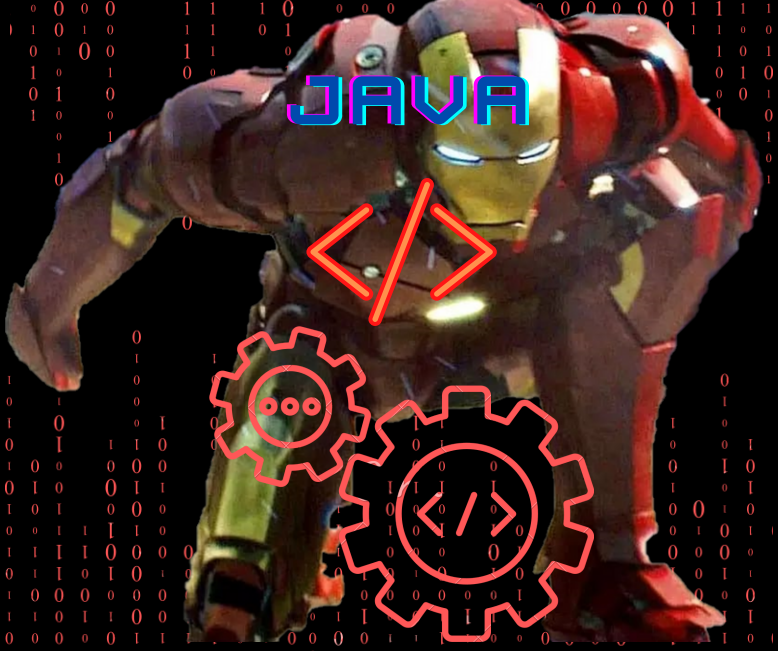
2023

Java

DSA
PART-1

CYBER TRON

JAVA



(THIRD PART)

T.I.M. HAMIM

Index: DSA (Part-1)

1.Intro to DSA

2.Arrays

3.ArrayList

4.Searching

5.Sorting

6.String

7.Huffman Coding

8.----------

9.----------

10.----------

11.----------

12.----------

2. Arrays

An array is a collection of variables of the same data type.

Types of array:

There are mainly two kinds of array :

- One dimensional (1-D) arrays or Linear arrays,
- Multi dimensional arrays
  (a) Two dimensional (2-D) arrays or Matrix arrays,
  (b) Three dimensional (3-D) arrays.

- **Declaring and creating 1-D array:**

declaring variables for storing 10 numbers:

```
int num1, num2, num3, num4, num5, num6,
num7, ..... , num10
```

Using array:

```
int[ ] num = new int[10];
```

Declaring array:

```
datatype[ ] array_name;
```

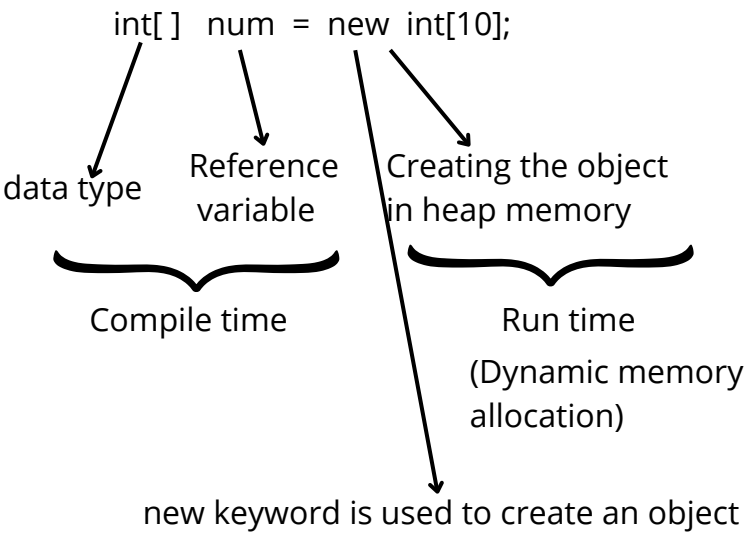
Example:

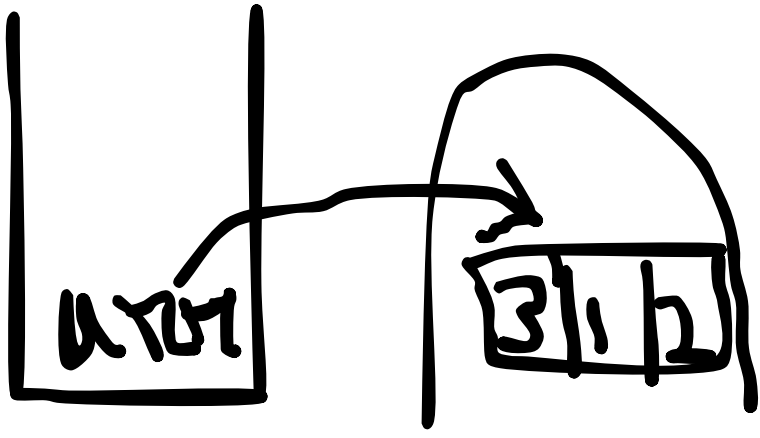
```
int[ ] number;
string[ ] names;
```

Creating array:

For creating an array you can use new operator.

Here,


int[ ] num = new int[10];
data type
Reference variable
Creating the object in heap memory
Compile time
Run time
(Dynamic memory allocation)
new keyword is used to create an object



Stack memory Heap memory

1. Array object are in heap.
2. Heap object are not continuous.
3. DMA in heap section during run time.
4. Array objects in Java may not be continuous as like C/C++. Because it depends on JVM.

Example:

```
 int[ ]  number;   //declaration
 number = new int[10]; //creation
```

or,

```
 int[ ]  number = new int[10];  //declaration and creation
```

or,

```
 int num1[ ];
 num1 = new int[5];
```

or,

```
  int[ ] num2 = new int[5];
```

- **What is the output of the following Java program fragment:**

```
package basicjava;
import java.util.Scanner;

public class JavaCode{
   public static void main(String[] args)
   {
      Scanner scan = new Scanner(System.in);
      int[] number1 = new int[5], number2 = new int[10];
      int num1[], num2[];
      num1 = new int[5];
      num2 = new int[5];
      number1[0] = 10;
      number1[1] = 11;
      number1[2] = 12;
      number1[3] = 13;
      number1[4] = 14;
      number2[0] = 15;
      number2[1] = 16;
      num1[0] = 1;
      num1[1] = 2;
      num2[0] = 3;
      num2[1] = 4;
      System.out.println("number1[0] = "+number1[0]);
      System.out.println("number1[1] = "+number1[1]);
      System.out.println("number1[2] = "+number1[2]);
      System.out.println("number1[3] = "+number1[3]);
```

```
        System.out.println("number1[4] = "+number1[4]);
        System.out.println("number2[0] = "+number2[0]);
        System.out.println("number2[1] = "+number2[1]);
        System.out.println("num1[0] = "+num1[0]);
        System.out.println("num1[1] = "+num1[1]);
        System.out.println("num2[0] = "+num2[0]);
        System.out.println("num2[1] = "+num2[1]);
    }
}
```

Output:

```
number1[0] = 10
number1[1] = 11
number1[2] = 12
number1[3] = 13
number1[4] = 14
number2[0] = 15
number2[1] = 16
num1[0] = 1
num1[1] = 2
num2[0] = 3
num2[1] = 4
```

- **Make a program that will be able to print the array length.**

```
package basicjava;
import java.util.Scanner;

public class JavaCode{
   public static void main(String[] args)
   {
      Scanner scan = new Scanner(System.in);
      int[] number = new int[5];

      number[0] = 10;
      number[1] = 11;
      number[2] = 12;
      number[3] = 13;
      number[4] = 14;
      int len = number.length;
      System.out.println("Length of array is "+len);
   }
}
```

Output:
Length of array is 5

- **What is the output of the following Java program fragment:**

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

public class Test {
   public static void main(String[] args) {
      Scanner scan = new Scanner(System.in);

      String[] str = new String[4];

      for(int i=0; i<str.length; i++){
         str[i] = scan.next();
      }

      System.out.println(Arrays.toString(str));
   }
}
```

Output:

```
1 2 3 4
[1, 2, 3, 4]
```

Here,
Arrays.toString(String argument) function/method prints all the value as String one by one using comma(,).

- **What will be the output of the program given bellow.**

```
import java.util.Arrays;

public class Test {
   public static void main(String[] args) {
      int[] arr = {1, 3, 2, 45, 6};
      change(arr);
      System.out.println(Arrays.toString(arr));
   }

   static void change(int[] nums) {
      nums[0] = 99;
   }
}
```

Output:
[99, 3, 2, 45, 6]

Here,
In Array when we pass the array in the change() function nums[ ] array actually pointing to the heap memory address of these array index and if we change any of value then it will be changed for both. That's why the arr[ ] got changed.
But it is actually call by value.

- **What is the output of the following Java program fragment:**

```
package basicjava;
import java.util.Scanner;

public class JavaCode{
   public static void main(String[] args)
   {
      Scanner scan = new Scanner(System.in);
      double[] number = new double[5];
      System.out.println("Please enter 5 numbers :
");
      for (int i = 0; i < 5; i++) {
         System.out.print("number ["+i+"] = ");
         number[i] = scan.nextDouble();
      }
      System.out.println("All numbers are :");
      for (int i = 0; i < 5; i++) {
         System.out.println("number ["+i+"] =
"+number[i]);
      }
   }
}
```

Output:

```
Please enter 5 numbers :
number [0] = 12.5
number [1] = 6
number [2] = 7
number [3] = 90
number [4] = 7
```

```
All numbers are :
number [0] = 12.5
number [1] = 6.0
number [2] = 7.0
number [3] = 90.0
number [4] = 7.0
```

- **What is the output of the following Java program fragment:**

```
package basicjava;
import java.util.Scanner;

public class JavaCode{
   public static void main(String[] args)
   {
      Scanner scan = new Scanner(System.in);
      int n;
      System.out.print("Enter num: ");
      n = scan.nextInt();
      double[] number = new double[n];
      System.out.println("Please enter "+n+" numbers : ");
      for (int i = 0; i < n; i++) {
         System.out.print("number ["+i+"] = ");
         number[i] = scan.nextDouble();
      }
      System.out.println("All numbers are :");
      for (int i = 0; i < n; i++) {
         System.out.println("number ["+i+"] = "+number[i]);
      }
   }
}
```

Output:
Enter num: 7
Please enter 7 numbers :
number [0] = 1
number [1] = 2
number [2] = 3
number [3] = 4
number [4] = 5
number [5] = 6
number [6] = 7
All numbers are :
number [0] = 1.0
number [1] = 2.0
number [2] = 3.0
number [3] = 4.0
number [4] = 5.0
number [5] = 6.0
number [6] = 7.0

Output:

```
Please enter 5 numbers :
number [0] = 1
number [1] = 2
number [2] = 3
number [3] = 4
number [4] = 5
Sum = 15.0
Average = 3.0
```

- **Make a program that will take 5 numbers and print the sum, average, minimum and maximum number of all numbers using array.**

```
package basicjava;

import java.util.Scanner;

public class JavaCode {

    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        double[] number = new double[5];
        double sum = 0, avg=0, min=0, max=0;
        System.out.println("Please enter 5 numbers : ");
        for (int i = 0; i < 5; i++) {
            System.out.print("number [" + i + "] = ");
            number[i] = scan.nextDouble();
            if(i==0){
                min = number[0];
            }
```

```
            if(min>number[i]){
                min = number[i];
            }
            else if(max<number[i]){
                max = number[i];
            }
            sum = sum + number[i];
        }
        System.out.println("Sum = " + sum);
        System.out.println("Average = " + (avg = sum / 5));
        System.out.println("Minimum number = "+min);
        System.out.println("Maximum number = "+max);
    }
}
```

Output:

```
Please enter 5 numbers :
number [0] = 2
number [1] = 4
number [2] = 1
number [3] = 5
number [4] = 3
Sum = 15.0
Average = 3.0
Minimum number = 1.0
Maximum number = 5.0
```

- **What is the output of the following Java program fragment:**

```
package basicjava;
import java.util.Scanner;

public class JavaCode {

   public static void main(String[] args) {
      Scanner scan = new Scanner(System.in);
      String[] names = new String[4];
      names[0] = "Hamim Talukder";
      names[1] = "Jim";
      names[2] = "Rahim";
      names[3] = "Mithu";
      for (int i = 0; i < 4; i++) {
         System.out.println(names[i]);
      }
      System.out.println("String length : "+
(names.length));
   }
}
```

or,

```
package basicjava;
import java.util.Scanner;
public class JavaCode {
   public static void main(String[] args) {
      Scanner scan = new Scanner(System.in);
      String names[ ] = new String[4];
      names[0] = "Hamim Talukder";
      names[1] = "Jim";
      names[2] = "Rahim";
```

```
        names[3] = "Mithu";
        for (int i = 0; i < 4; i++) {
            System.out.println(names[i]);
        }
        System.out.println("String length : "+(names.length));
    }
}
```

or,

```
package basicjava;
import java.util.Scanner;

public class JavaCode {

    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        String[] names = {"Hamim
Talukder","Jim","Rahim","Mithu"};
 for (int i = 0; i < 4; i++) {
 System.out.println(names[i]);
 }
 System.out.println("String length : "+
(names.length));
  }
}
```

Output:

```
Hamim Talukder
Jim
Rahim
Mithu
String length : 4
```

- **What is the output of the following Java program fragment:**

```
package basicjava;
import java.util.Scanner;

public class JavaCode {

   public static void main(String[] args) {
      Scanner scan = new Scanner(System.in);
      String[] names = {"Hamim
Talukder","Jim","Rahim","Mithu"};
      for (String x : names) {
         System.out.println(x);
      }
   }
}
```

Output:
Hamim Talukder
Jim
Rahim
Mithu

Here,
We have used "for each loop"/"enhanced loop". Here all the values of names string array will be assigned one by one in x variable and then it will be printed.

- **What is the output of the following Java program fragment:**

```
package basicjava;
import java.util.Scanner;

public class JavaCode {
   public static void main(String[] args) {
      Scanner scan = new Scanner(System.in);
      int[] num = {10,20,30,40,50,60,70,80,90,100};
      int sum = 0;
      for (int x : num) {
         System.out.println(x);
         sum = sum + x;
      }
      System.out.println("sum = "+sum);
   }
}
```

Output:

```
10
20
30
40
50
60
70
80
90
100
sum = 550
```

- **Deciaring and creating 2-D array:**

```
int[ ][ ] number = new int[2][3];
                       /    \
                     row    column
```

Total elements = rows * column = 2 * 3 = 6

or,
int number[ ][ ];
number = new int[n][m];
or,
number = new int[n][ ];

or,
int numb

	Column 0	Column 1	Column 2
Row 0	x[0][0]	x[0][1]	x[0][2]
Row 1	x[1][0]	x[1][1]	x[1][2]
Row 2	x[2][0]	x[2][1]	x[2][2]

- **What is the output of the following Java program fragment:**

```
import java.util.Scanner;
public class main{

   public static void main(String[] args)
   {
      Scanner scan = new Scanner(System.in);
      int num1[][], n,m;

      System.out.print("Enter array row and culamn
size: ");
      n=scan.nextInt();
      m=scan.nextInt();

      num1 = new int[n][m];

      for(int i=0; i<n;i++){
         for(int j=0;j<m;j++){
            num1[i][j]=scan.nextInt();
         }
      }
      System.out.print("\n");

      for(int i=0; i<n;i++){
         for(int j=0;j<m;j++){
            System.out.println(num1[i][j]);
         }
      }
   }
}
```

Output:

```
Enter array row and culamn size: 2 3
1 2 3
4 5 6

1
2
3
4
5
6
```

- **What is the output of the following Java program fragment:**

```
package basicjava;
import java.util.Scanner;
```

```
public class JavaCode {
   public static void main(String[] args) {
      Scanner scan = new Scanner(System.in);
      int[][] num = new int[2][3];
      for (int i = 0; i < 2; i++) {
         for (int j = 0; j < 3; j++) {
            num[i][j] = scan.nextInt();
         }
      }
      for (int i = 0; i < 2; i++) {
         for (int j = 0; j < 3; j++) {
            System.out.println("num["+i+"]["+j+"] =
"+num[i][j]);
         }
     }
  }
}
```

Output:

```
1
2
3
4
5
6
num[0][0] = 1
num[0][1] = 2
num[0][2] = 3
num[1][0] = 4
num[1][1] = 5
num[1][2] = 6
```

- **What is the output of the following Java program fragment:**

```
public class Test {
  public static void main(String[] args) {
    int[][] arr = {
        { 1, 3, 2 },
        { 4, 5 },
        { 6, 7, 8, 9 }
    };

    System.out.println("Printing value : ");
    for (int i = 0; i < arr.length; i++) {
      for (int j = 0; j < arr[i].length; j++) {
        System.out.print(arr[i][j]+" ");
      }
      System.out.println();
    }
  }

}
```

Output:

Printing value :
1 3 2
4 5
6 7 8 9

- **What is the output of the following Java program fragment:**

```
import java.util.Scanner;

public class Test {
  public static void main(String[] args) {
    Scanner scan = new Scanner (System.in);

    int[][] arr = new int[3][3];

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

    System.out.println("Printing value : ");
    for(int i=0; i<arr.length; i++){
      for(int j=0; j<arr[i].length; j++){
        System.out.print(arr[i][j]+" ");
      }
      System.out.println();
    }
  }
}
```

Output:

```
1 2 3 4 5 6 7 8 9
Printing value :
1 2 3
4 5 6
7 8 9
```

- **Make a program that will show sum of A matrix and B matrix using array.**

```
package basicjava;

import java.util.Scanner;

public class JavaCode {

    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        int n, m, i, j;
        System.out.print("Enter the row : ");
        n = scan.nextInt();
        System.out.print("Enter the Column : ");
        m = scan.nextInt();
        int[][] A = new int[n][m], B = new int[n][m], C =
new int[n][m];
        System.out.println();
        //Scaning A matrix
        System.out.printf("Enter elements for A matrix
:\n\n");
        for (i = 0; i < n; i++) {
            for (j = 0; j < m; j++) {
                System.out.print("A[" + i + "][" + j + "] = ");
                A[i][j] = scan.nextInt();
            }
            System.out.println();
        }
        //Printing A matrix
        System.out.print("A=");
        for (i = 0; i < n; i++) {
```

```
            System.out.printf("\t");
            for (j = 0; j < m; j++) {
                System.out.printf("%d ", A[i][j]);
            }
            System.out.printf("\n\n");
        }
        //Scaning B matrix
        System.out.printf("\nEnter elements for B
matrix :\n\n");
        for (i = 0; i < n; i++) {
            for (j = 0; j < m; j++) {
                System.out.printf("B[%d][%d]=", i, j);
                B[i][j] = scan.nextInt();
            }
            System.out.println();
        }
        //Printing B matrix
        System.out.print("B=");
        for (i = 0; i < n; i++) {
            System.out.printf("\t");
            for (j = 0; j < m; j++) {
                System.out.printf("%d ", B[i][j]);
            }
            System.out.printf("\n\n");
        }
         //Adding the matrix
        System.out.printf("A+B = ");
        for (i = 0; i < n; i++) {
            System.out.printf("\t");
            for (j = 0; j < m; j++) {
                C[i][j] = A[i][j] + B[i][j];
```

```
 System.out.print(C[i][j] + " ");
 }
 System.out.printf("\n\n");
 }
 System.out.println();
 System.out.println();
 }
}
```

- **Make a program that will show subtraction of A matrix and B matrix using array.**

```
package basicjava;

import java.util.Scanner;

public class JavaCode {

    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        int n, m, i, j;
        System.out.print("Enter the row : ");
        n = scan.nextInt();
        System.out.print("Enter the Column : ");
        m = scan.nextInt();
        int[][] A = new int[n][m], B = new int[n][m], C =
new int[n][m];
        System.out.println();
        //Scaning A matrix
        System.out.printf("Enter elements for A matrix
:\n\n");
        for (i = 0; i < n; i++) {
            for (j = 0; j < m; j++) {
                System.out.print("A[" + i + "][" + j + "] = ");
                A[i][j] = scan.nextInt();
            }
            System.out.println();
        }
        //Printing A matrix
        System.out.print("A=");
        for (i = 0; i < n; i++) {
```

```
            System.out.printf("\t");
            for (j = 0; j < m; j++) {
                System.out.printf("%d ", A[i][j]);
            }
            System.out.printf("\n\n");
        }
        //Scaning B matrix
        System.out.printf("\nEnter elements for B
matrix :\n\n");
        for (i = 0; i < n; i++) {
            for (j = 0; j < m; j++) {
                System.out.printf("B[%d][%d]=", i, j);
                B[i][j] = scan.nextInt();
            }
            System.out.println();
        }
        //Printing B matrix
        System.out.print("B=");
        for (i = 0; i < n; i++) {
            System.out.printf("\t");
            for (j = 0; j < m; j++) {
                System.out.printf("%d ", B[i][j]);
            }
            System.out.printf("\n\n");
        }
        //Adding the matrix
        System.out.printf("A-B = ");
        for (i = 0; i < n; i++) {
            System.out.printf("\t");
            for (j = 0; j < m; j++) {
                C[i][j] = A[i][j] - B[i][j];
```

```
            System.out.print(C[i][j] + " ");
          }
          System.out.println();
          System.out.println();
        }
    }
}
```

- **Make a program that will show multiplication of two matrix A and B using array.**

```
package basicjava;

import java.util.Scanner;

public class JavaCode {

    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        int r1, c1, r2, c2, i, j, k, sum = 0;
        System.out.printf("\nEnter the Raw and
Column for A matrix :");
        r1 = scan.nextInt();
        c1 = scan.nextInt();
        System.out.printf("\nEnter the Raw and
Column for B matrix :");
        r2 = scan.nextInt();
        c2 = scan.nextInt();
        while (c1 != r2) {
```

```
            System.out.printf("\nError!! column of first
matrix is not equal to raw of second matrix\n");

            System.out.printf("\nEnter the Raw and
Column for A matrix :");
            r1 = scan.nextInt();
            c1 = scan.nextInt();
            System.out.printf("\nEnter the Raw and
Column for B matrix :");
            r2 = scan.nextInt();
            c2 = scan.nextInt();
        }
        int[][] A = new int[r1][c1], B = new int[r2][c2], C
= new int[r1][c2];
        System.out.printf("\n");
        // Scaning A matrix
        System.out.printf("Enter elements for A matrix
:\n\n");
        for (i = 0; i < r1; i++) {
            for (j = 0; j < c1; j++) {
                System.out.printf("A[%d][%d]=", i, j);
                A[i][j] = scan.nextInt();
            }
            System.out.printf("\n");
        }
        // Scaning B matrix
        System.out.printf("\nEnter elements for B
matrix :\n\n");
        for (i = 0; i < r2; i++) {
            for (j = 0; j < c2; j++) {
                System.out.printf("B[%d][%d]=", i, j);
                B[i][j] = scan.nextInt();
```

```
  }
   System.out.printf("\n");
}
// Printing A matrix
System.out.printf("A=");
for (i = 0; i < r1; i++) {
   System.out.printf("\t");
   for (j = 0; j < c1; j++) {
      System.out.printf("%d ", A[i][j]);
   }
   System.out.printf("\n\n");
}
// Printing B matrix
System.out.printf("\nB=");
for (i = 0; i < r2; i++) {
   System.out.printf("\t");
   for (j = 0; j < c2; j++) {
      System.out.printf("%d ", B[i][j]);
   }
   System.out.printf("\n\n");
}
//Multiplying matrix
System.out.printf("A*B = ");
for (i = 0; i < r1; i++) {
   System.out.printf("\t");
   for (j = 0; j < c2; j++) {
      for (k = 0; k < c1; k++) {
         sum = sum + A[i][k] * B[k][j];
      }
      C[i][j] = sum;
      sum = 0;
```

```
            System.out.printf("%d ", C[i][j]);
          }
          System.out.printf("\n\n");
       }
    }
}
```

- **Make a program that will show upper and lower triangle elements of A matrix using array.**

```
package basicjava;

import java.util.Scanner;

public class JavaCode {

    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        int n, m, i, j, uppersum = 0, lowersum = 0;
        System.out.print("Enter the row : ");
        n = scan.nextInt();
        System.out.print("Enter the Column : ");
        m = scan.nextInt();
        int[][] A = new int[n][m];
        System.out.println();
        //Scaning A matrix
        System.out.printf("Enter elements for A matrix
:\n\n");
        for (i = 0; i < n; i++) {
            for (j = 0; j < m; j++) {
                A[i][j] = scan.nextInt();
            }
```

```
        }
        //Printing A matrix
        System.out.print("A=");
        for (i = 0; i < n; i++) {
            System.out.printf("\t");
            for (j = 0; j < m; j++) {
                System.out.printf("%d ", A[i][j]);
            }
            System.out.printf("\n\n");
        }
        //Adding upper trainguller matrix
        for (i = 0; i < n; i++) {
            for (j = 0; j < m; j++) {
                if (i < j) {
                    uppersum = uppersum + A[i][j];
                }
                if (i > j) {
                    lowersum = lowersum + A[i][j];
                }
                System.out.println();
            }
        }
        System.out.printf("Sum of upper triangle
elements is %d", uppersum);
        System.out.printf("\nSum of lower triangle
elements is %d", lowersum);
    }
}
```

- **Make a program that will print given bellow.**
  0 1 2 3 4
  5 6 7 8 9
  10 11 12 13 14
  15 16 17 18 19

```
package basicjava;

import java.util.Scanner;

public class JavaCode {

    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        int[][] num = new int[4][5];
        int k=0;
        //assigning the value to the 2d array
        for (int i = 0; i < 4; i++) {
            for (int j = 0; j < 5; j++) {
                num[i][j] = k;
                k++;
            }
        }
        for (int i = 0; i < 4; i++) {
            for (int j = 0; j < 5; j++) {
                System.out.print(num[i][j]+" ");
            }
            System.out.println();
        }
    }
}
```

- **Make a program that will print given bellow.**

  0

  1 2

  3 4 5

  6 7 8 9

```
package basicjava;
import java.util.Scanner;
public class JavaCode {
   public static void main(String[] args) {
      Scanner scan = new Scanner(System.in);
      int[][] num = new int[4][];
      // declaring column length one by one
      num[0] = new int[1];
      num[1] = new int[2];
      num[2] = new int[3];
      num[3] = new int[4];
      int k=0;
      //assigning the value to the 2d array
      for (int i = 0; i < 4; i++) {
         for (int j = 0; j < i+1; j++) {
            num[i][j] = k;
            k++;
         }
      }
      for (int i = 0; i < 4; i++) {
         for (int j = 0; j < i+1; j++) {
            System.out.print(num[i][j]+" ");
         }
         System.out.println();
      }
   }
}
```

or,

```
package basicjava;
import java.util.Scanner;

public class JavaCode {
   public static void main(String[] args) {
      Scanner scan = new Scanner(System.in);
      int[][] num = new int[4][];
      // declaring column length one by one
      for (int i = 0; i < 4 ; i++) {
         num[i] = new int[i+1];
      }
      int k=0;
      //assigning the value to the 2d array
      for (int i = 0; i < 4; i++) {
         for (int j = 0; j < i+1; j++) {
            num[i][j] = k;
            k++;
         }
      }
      for (int i = 0; i < 4; i++) {
         for (int j = 0; j < i+1; j++) {
            System.out.print(num[i][j]+" ");
         }
         System.out.println();
      }
   }
}
```

- **Make a program that will sort integer array elements in ascending and descending.**

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

public class JavaCode {
   public static void main(String[] args) {
      Scanner scan = new Scanner(System.in);
      int[] number = {10, -3, 18, 5,25};
      Arrays.sort(number);
      System.out.print("Ascending :");
      for (int i = 0; i < 5; i++) {
         System.out.print (" "+number[i]);
      }
      System.out.println();
      System.out.print("Descending :");
      for (int i = 4; i >=0; i--) {
         System.out.print(" "+number[i]);
      }
   }
}
```

Output:
Ascending : -3 5 10 18 25
Descending : 25 18 10 5 -3

Here,
We have used Arrays class and sort method (Arrays.sort) to sort the arrays elements.

- **Make a program that will sort String array elements in ascending and descending.**

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

public class JavaCode {
   public static void main(String[] args) {
      Scanner scan = new Scanner(System.in);
      String[] names = {"Hamim Talukder", "Jim",
"Rahim", "Mithu vi"};
      Arrays.sort(names);
      System.out.print("Ascending :");
      for (int i = 0; i < 4; i++) {
         System.out.print ("  "+names[i]);
      }
      System.out.println();
      System.out.print("Descending :");
      for (int i = 3; i >=0; i--) {
         System.out.print("  "+names[i]);
      }
   }
}
```

Output:
Ascending : Hamim Talukder Jim Mithu vi Rahim
Descending : Rahim Mithu vi Jim Hamim Talukder

3. ArrayList

The ArrayList class is a resizable array, which can be found in the java.util package.

The difference between a built-in array and an ArrayList in Java, is that the size of an array cannot be modified (if you want to add or remove elements to/from an array, you have to create a new one). While elements can be added and removed from an ArrayList whenever you want.

Syntax:

```
ArrayList<Integer> number = new ArrayList<Integer>();
```

1. ArrayList class uses a dynamic array for sorting the elements.
2. ArrayList is better for sorting and accessing data.
3. Slow for manipulating data (deleting or inserting data)
4. Can contain duplicate elements.

- **Custom built ArrayList:**

- **Make a custom built integer type ArrayList manually:**

```
package List;

import java.util.Arrays;

public class CustomArrayList {
    private int[] data;
    private static int DEFAULT_SIZE = 10;
    private int size = 0; // also working as index value

    public CustomArrayList(){
        this.data = new int[DEFAULT_SIZE];
    }

    public void add(int num){
        if(isFull()){
            resize();
        }
        data[size++] = num;
    }

    private void resize() {
        int[] temp = new int[data.length * 2];

        // copy the current itemps in the new array
        for(int i = 0; i < data.length; i++){
            temp[i] = data[i];
```

```
    }
    data = temp;
}

private boolean isFull() {
    return size == data.length;
}

public int remove(){
    int removed = data[--size];
    return removed;
}

public int get(int index){
    return data[index];
}

public int size(){
    return size;
}

public void set(int index, int value){
    data[index] = value;
}

@Override
public String toString(){
    return "CustomArrayList{" +
            "data=" + Arrays.toString(data) +
            ", size=" + size +
            '}';
```

```
    }

    public static void main(String[] args) {
        CustomArrayList list = new CustomArrayList();
        list.add(3);
        list.add(5);
        list.add(9);

        System.out.println(list);

        for(int i = 0; i < 14; i++){
            list.add(2 * i);
        }

        System.out.println(list);
    }
}
```

Output:

```
CustomArrayList{data=[3, 5, 9, 0, 0, 0, 0, 0, 0, 0], size=3}
CustomArrayList{data=[3, 5, 9, 0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 0, 0, 0], size=17}
```

- **Make a custom built multitype ArrayList using Generics:**

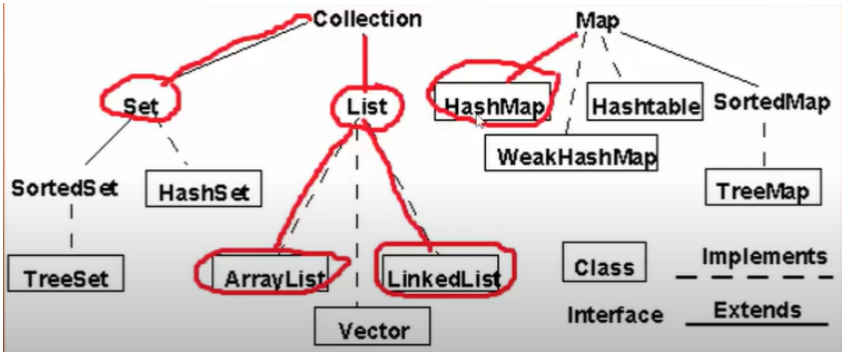
```
package List;

import java.util.Arrays;

public class CustomArrayList<T> {
  private Object[] data;
  private static int DEFAULT_SIZE = 10;
  private int size = 0; // also working as index value

  public CustomArrayList(){
    this.data = new Object[DEFAULT_SIZE];
  }

  public void add(T num){
    if(isFull()){
      resize();
    }
    data[size++] = num;
  }

  private void resize() {
    Object[] temp = new Object[data.length * 2];

    // copy the current itemps in the new array
    for(int i = 0; i < data.length; i++){
      temp[i] = data[i];
    }
    data = temp;
  }
```

```
private boolean isFull() {
  return size == data.length;
}

public T remove(){
  T removed = (T) data[--size];
  return removed;
}

public T get(int index){
  return (T) data[index];
}

public int size(){
  return size;
}

public void set(int index, T value){
  data[index] = value;
}

@Override
public String toString(){
  return "CustomArrayList{" +
      "data=" + Arrays.toString(data) +
      ", size=" + size +
      '}';
}
```

```
  public static void main(String[] args) {
    CustomArrayList<Integer> list = new
CustomArrayList<>();
    list.add(3);
    list.add(5);
    list.add(9);

    System.out.println(list);

    for(int i = 0; i < 14; i++){
      list.add(2 * i);
    }

    System.out.println(list);
  }
}
```

Output:

CustomArrayList{data=[3, 5, 9, null, null, null, null, null, null, null], size=3}
CustomArrayList{data=[3, 5, 9, 0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, null, null, null], size=17}


Collection
Map
Set
List
HashMap
Hashtable
SortedMap
WeakHashMap
SortedSet
HashSet
TreeMap
TreeSet
ArrayList
LinkedList
Class
Implements
Interface
Extends
Vector

Array vs ArrayList :

ArrayList is a part of the Java collection framework and it is a class of java.util package. It provides us with dynamic arrays in Java. Though, it may be slower than standard arrays but can be helpful in programs where lots of manipulation in the array is needed. This class is found in java.util package. The main advantages of ArrayList are, if we declare an array then it's needed to mention the size but in ArrayList, it is not needed to mention the size of ArrayList if you want to mention the size then you can do it.

Array	ArrayList
1.Not Resizable	1.Resizable
2.for, for each loop	2.for, for each loop, iterator
3.Fast	3.Slow
4.array.length	4.array.size()

- **Inbuilt default ArrayList:**

Some methods related to ArrayList :

- **add(element)** : Adds an element to the end of the ArrayList.
- **add(index, element)** : Inserts an element at the specified index in the ArrayList.
- **addAll(collection)** : Adds all elements from a specified collection to the end of the ArrayList.
- **addAll(index, collection)** : Inserts all elements from a specified collection into the ArrayList, starting at the specified index.
- **clear()** : Removes all elements from the ArrayList.
- **clone()** : Creates a shallow copy of the ArrayList.
- **contains(element)** : Checks if the ArrayList contains a specific element.
- **equals( )** :
- **get(index)** : Retrieves the element at the specified index in the ArrayList.
- **indexOf(element)** : Returns the index of the first occurrence of the specified element in the ArrayList, or -1 if not found.
- **isEmpty()** : Checks if the ArrayList is empty.
- **iterator()** : Returns an iterator to iterate over the elements in the ArrayList.
- **remove(index)** : Removes the element at the specified index from the ArrayList.

- **remove(element) :** Removes the first occurrence of the specified element from the ArrayList, if present.
- **removeAll(collection)** : Removes all elements in the ArrayList that are also contained in the specified collection.
- **retainAll(collection)** : Removes all elements in the ArrayList that are not contained in the specified collection.
- **set(index, element)** : Replaces the element at the specified index in the ArrayList with the specified element.
- **size()** : Returns the number of elements in the ArrayList.
- **subList(fromIndex, toIndex)** : Returns a new ArrayList containing elements from the specified fromIndex (inclusive) to the specified toIndex (exclusive).
- **toArray()** : Converts the ArrayList to an array.
- **toString()** : Returns a string representation of the ArrayList.
- **ensureCapacity(minCapacity):** Increases the capacity of the list to ensure it can hold at
- **lastIndexOf(element):** Returns the index of the last occurrence of the specified element.
- **listIterator():** Returns a list iterator over the elements in the list.
- **listIterator(index):** Returns a list iterator over the elements in the list, starting at the specified index.

- **replaceAll(operator):** Replaces each element of the list with the result of applying the specified operator.
- **sort(comparator):** Sorts the elements of the list according to the specified comparator.
- **trimToSize():** Trims the capacity of the list to the current size.

- **add(element) :**

- **What is the output of the following Java program fragment:**

```
import java.util.ArrayList;

public class Test {
   public static void main(String[] args) {
      ArrayList<String> names = new ArrayList<>();
      names.add("John");
      names.add("Alice");
      names.add("Bob");

      System.out.println(names);
   }
}
```

Output:
[John, Alice, Bob]

- **add(index, element) :**

- **What is the output of the following Java program fragment:**

```
import java.util.ArrayList;

public class Test {
   public static void main(String[] args) {
      ArrayList<String> colors = new ArrayList<>();
      colors.add("Red");
      colors.add("Blue");
      colors.add("Green");

      colors.add(1, "Yellow");

      System.out.println(colors);
   }
}
```

Output:
[Red, Yellow, Blue, Green]

- **addAll(collection) :**

- **What is the output of the following Java program fragment:**

```
import java.util.ArrayList;
import java.util.Arrays;

public class Test {
   public static void main(String[] args) {
      ArrayList<Integer> numbers = new ArrayList<>();
      numbers.add(1);
      numbers.add(2);
      numbers.add(3);

      ArrayList<Integer> moreNumbers = new ArrayList<>
(Arrays.asList(4, 5, 6));
      numbers.addAll(moreNumbers);

      System.out.println(numbers);
   }
}
```

Output:
[1, 2, 3, 4, 5, 6]

- **addAll(index, collection) :**

- **What is the output of the following Java program fragment:**

```
import java.util.ArrayList;
import java.util.Arrays;

public class Test {
   public static void main(String[] args) {
      ArrayList<String> fruits = new ArrayList<>
(Arrays.asList("Apple", "Orange", "Banana"));

      ArrayList<String> additionalFruits = new ArrayList<>
(Arrays.asList("Mango", "Grapes"));
      fruits.addAll(1, additionalFruits);

      System.out.println(fruits);
   }
}
```

Output:
[Apple, Mango, Grapes, Orange, Banana]

- **clear() :**

- **What is the output of the following Java program fragment:**

```
import java.util.ArrayList;

public class Test {
   public static void main(String[] args) {
      ArrayList<String> fruits = new ArrayList<>();

      fruits.add("Apple");
      fruits.add("Banana");
      fruits.add("Orange");

      System.out.println("ArrayList before clearing: " +
fruits);

      fruits.clear();

      System.out.println("ArrayList after clearing: " + fruits);
   }
}
```

Output:
ArrayList before clearing: [Apple, Banana, Orange]
ArrayList after clearing: []

- **clone() :**

- **What is the output of the following Java program fragment:**

```
import java.util.ArrayList;

public class Test{
   public static void main(String[] args) {
      ArrayList<String> originalList = new ArrayList<>();

      originalList.add("Apple");
      originalList.add("Banana");
      originalList.add("Orange");

      ArrayList<String> clonedList = (ArrayList<String>)
originalList.clone();

      clonedList.add("Mango");

      System.out.println("Original List: " + originalList);
      System.out.println("Cloned List: " + clonedList);
   }
}
```

Output:
Original List: [Apple, Banana, Orange]
Cloned List: [Apple, Banana, Orange, Mango]

- **What is the output of the following Java program fragment:**

```
import java.util.ArrayList;

public class Test {
    public static void main(String[] args) {
        ArrayList<Person> originalList = new ArrayList<>();

        originalList.add(new Person("John", 25));
        originalList.add(new Person("Alice", 30));
        originalList.add(new Person("Bob", 28));

        ArrayList<Person> clonedList = (ArrayList<Person>)
originalList.clone();

        clonedList.get(0).setAge(26);

        System.out.println("Original List: " + originalList);
        System.out.println("Cloned List: " + clonedList);
    }

    static class Person {
        private String name;
        private int age;

        public Person(String name, int age) {
            this.name = name;
            this.age = age;
        }
```

```
    public String getName() {
        return name;
    }

    public int getAge() {
        return age;
    }

    public void setAge(int age) {
        this.age = age;
    }

    @Override
    public String toString() {
        return "Person [name=" + name + ", age=" + age +
"]";
    }
  }
}
```

Output:

Original List: [Person [name=John, age=26], Person [name=Alice, age=30], Person [name=Bob, age=28]]

Cloned List: [Person [name=John, age=26], Person [name=Alice, age=30], Person [name=Bob, age=28]]

- **What is the output of the following Java program fragment:**

```
import java.util.ArrayList;

public class Test {
    public static void main(String[] args) {
        ArrayList<String> colors = new ArrayList<>();

        colors.add("Red");
        colors.add("Green");
        colors.add("Blue");

        boolean containsGreen = colors.contains("Green");
        boolean containsYellow = colors.contains("Yellow");

        System.out.println("Does the ArrayList contain
'Green'? " + containsGreen);
        System.out.println("Does the ArrayList contain
'Yellow'? " + containsYellow);
    }
}
```

Output:

```
Does the ArrayList contain 'Green'? true
Does the ArrayList contain 'Yellow'? false
```

- **equals( ) :**
- **What is the output of the following Java program fragment:**

```
import java.util.ArrayList;

public class Test {
   public static void main(String[] args) {
      ArrayList<String> list1 = new ArrayList<>();

      list1.add("Apple");
      list1.add("Banana");
      list1.add("Orange");

      ArrayList<String> list2 = new ArrayList<>();

      list2.add("Apple");
      list2.add("Banana");
      list2.add("Orange");

      boolean areEqual = list1.equals(list2);

      System.out.println("Are the ArrayLists equal? " +
areEqual);
   }
}
```

Output:
Are the ArrayLists equal? true

- **get(index) :**

- **What is the output of the following Java program fragment:**

```
import java.util.ArrayList;

public class Test {
   public static void main(String[] args) {
      ArrayList<String> colors = new ArrayList<>();

      colors.add("Red");
      colors.add("Green");
      colors.add("Blue");

      String firstColor = colors.get(0);
      String secondColor = colors.get(1);
      String thirdColor = colors.get(2);

      System.out.println("First Color: " + firstColor);
      System.out.println("Second Color: " + secondColor);
      System.out.println("Third Color: " + thirdColor);
   }
}
```

Output:
First Color: Red
Second Color: Green
Third Color: Blue

- **indexOf(element) :**

- **What is the output of the following Java program fragment:**

```
import java.util.ArrayList;

public class Test {
   public static void main(String[] args) {
      ArrayList<String> colors = new ArrayList<>();

      colors.add("Red");
      colors.add("Green");
      colors.add("Blue");
      colors.add("Green");

      int indexOfGreen = colors.indexOf("Green");
      int indexOfYellow = colors.indexOf("Yellow");

      System.out.println("Index of 'Green': " +
indexOfGreen);
      System.out.println("Index of 'Yellow': " +
indexOfYellow);
   }
}
```

Output:
Index of 'Green': 1
Index of 'Yellow': -1

- **isEmpty() :**
- **What is the output of the following Java program fragment:**

```
import java.util.ArrayList;

public class Test {
   public static void main(String[] args) {
      ArrayList<String> colors = new ArrayList<>();

      boolean isEmpty = colors.isEmpty();

      System.out.println("Is the ArrayList empty? " +
isEmpty);

      colors.add("Red");
      colors.add("Green");
      colors.add("Blue");

      isEmpty = colors.isEmpty();

      System.out.println("Is the ArrayList empty? " +
isEmpty);
   }
}
```

Output:

```
Is the ArrayList empty? true
Is the ArrayList empty? false
```

- **iterator() :**

- **What is the output of the following Java program fragment:**

```
import java.util.ArrayList;
import java.util.Iterator;

public class Test {
   public static void main(String[] args) {
      ArrayList<String> colors = new ArrayList<>();

      colors.add("Red");
      colors.add("Green");
      colors.add("Blue");

      Iterator<String> iterator = colors.iterator();

      while (iterator.hasNext()) {
         String color = iterator.next();
         System.out.println(color);
      }
   }
}
```

Output:
Red
Green
Blue

- **remove(index) :**

- **What is the output of the following Java program fragment:**

```
import java.util.ArrayList;

public class Test {
   public static void main(String[] args) {
      ArrayList<String> colors = new ArrayList<>();

      colors.add("Red");
      colors.add("Green");
      colors.add("Blue");

      System.out.println("Original ArrayList: " + colors);

      String removedColor = colors.remove(1);

      System.out.println("Removed Color: " +
removedColor);
      System.out.println("Updated ArrayList: " + colors);
   }
}
```

Output:
Original ArrayList: [Red, Green, Blue]
Removed Color: Green
Updated ArrayList: [Red, Blue]

- **remove(element) :**

- **What is the output of the following Java program fragment:**

```
import java.util.ArrayList;

public class Test {
   public static void main(String[] args) {
      ArrayList<String> colors = new ArrayList<>();

      colors.add("Red");
      colors.add("Green");
      colors.add("Blue");

      System.out.println("Original ArrayList: " + colors);

      boolean removed = colors.remove("Green");

      System.out.println("Element 'Green' removed: " +
removed);
      System.out.println("Updated ArrayList: " + colors);
   }
}
```

Output:
Original ArrayList: [Red, Green, Blue]
Element 'Green' removed: true
Updated ArrayList: [Red, Blue]

- **removeAll(collection) :**

- **What is the output of the following Java program fragment:**

```
import java.util.ArrayList;
import java.util.Arrays;

public class Test {
   public static void main(String[] args) {
      ArrayList<String> colors = new ArrayList<>();

      colors.add("Red");
      colors.add("Green");
      colors.add("Blue");
      colors.add("Yellow");
      colors.add("Orange");

      System.out.println("Original ArrayList: " + colors);

      ArrayList<String> elementsToRemove = new
ArrayList<>(Arrays.asList("Green", "Yellow", "Orange"));

      boolean removed =
colors.removeAll(elementsToRemove);

      System.out.println("Elements removed: " + removed);
      System.out.println("Updated ArrayList: " + colors);
   }
}
```

Output:

```
Original ArrayList: [Red, Green, Blue, Yellow, Orange]
Elements removed: true
Updated ArrayList: [Red, Blue]
```

- **retainAll(collection) :**

- **What is the output of the following Java program fragment:**

```
import java.util.ArrayList;
import java.util.Arrays;

public class Test {
   public static void main(String[] args) {
      ArrayList<String> colors = new ArrayList<>();

      colors.add("Red");
      colors.add("Green");
      colors.add("Blue");
      colors.add("Yellow");
      colors.add("Orange");

      System.out.println("Original ArrayList: " + colors);

      ArrayList<String> elementsToRetain = new
ArrayList<>(Arrays.asList("Green", "Yellow"));

      boolean retained =
colors.retainAll(elementsToRetain);

      System.out.println("Elements retained: " + retained);
      System.out.println("Updated ArrayList: " + colors);
   }
}
```

Output:

Original ArrayList: [Red, Green, Blue, Yellow, Orange]
Elements retained: true
Updated ArrayList: [Green, Yellow]

- **set(index, element) :**

- **What is the output of the following Java program fragment:**

```
import java.util.ArrayList;

public class Test {
   public static void main(String[] args) {
      ArrayList<String> colors = new ArrayList<>();

      colors.add("Red");
      colors.add("Green");
      colors.add("Blue");

      System.out.println("Original ArrayList: " + colors);

      colors.set(1, "Yellow");

      System.out.println("Updated ArrayList: " + colors);
   }
}
```

Output:
Original ArrayList: [Red, Green, Blue]
Updated ArrayList: [Red, Yellow, Blue]

- **size() :**

- **What is the output of the following Java program fragment:**

```
import java.util.ArrayList;

public class Test {
   public static void main(String[] args) {
      ArrayList<String> colors = new ArrayList<>();

      colors.add("Red");
      colors.add("Green");
      colors.add("Blue");

      int size = colors.size();

      System.out.println("Size of the ArrayList: " + size);
   }
}
```

Output:
Size of the ArrayList: 3

- **subList(fromIndex, toIndex) :**

- **What is the output of the following Java program fragment:**

```
import java.util.ArrayList;
import java.util.List;

public class Test {
   public static void main(String[] args) {
      ArrayList<String> colors = new ArrayList<>();

      colors.add("Red");
      colors.add("Green");
      colors.add("Blue");
      colors.add("Yellow");
      colors.add("Orange");

      System.out.println("Original ArrayList: " + colors);

      List<String> subList1 = colors.subList(1, 3);

      System.out.println("Sun-List2: "+subList1);
      System.out.println("Index 1 element of subList1 :
"+subList1.get(1));

      System.out.println();

      // Or,
      ArrayList<String> subList2 = new ArrayList<>
(colors.subList(2, 4));
```

```
        System.out.println("Sub-List1: " + subList2);
        System.out.println("Index 1 element of subList2 :
"+subList2.get(1));
    }
}
```

Output:

```
Original ArrayList: [Red, Green, Blue, Yellow, Orange]
Sun-List2: [Green, Blue]
Index 1 element of subList1 : Blue

Sub-List1: [Blue, Yellow]
Index 1 element of subList2 : Yellow
```

- **toArray() :**

- **What is the output of the following Java program fragment:**

```
import java.util.ArrayList;
import java.util.Arrays;

public class Test {
   public static void main(String[] args) {
      ArrayList<String> colorsList = new ArrayList<>();

      colorsList.add("Red");
      colorsList.add("Green");
      colorsList.add("Blue");

      String[] colorsArray = colorsList.toArray(new
String[0]);

      System.out.println("Array: " +
Arrays.toString(colorsArray));

      System.out.println("Index 1 String :
"+colorsArray[1]);
   }
}
```

Output:
Array: [Red, Green, Blue]
Index 1 String : Green

- **toString() :**

- **What is the output of the following Java program fragment:**

```
import java.util.ArrayList;

public class Test {
   public static void main(String[] args) {
      ArrayList<String> colorsList = new ArrayList<>();

      colorsList.add("Red");
      colorsList.add("Green");
      colorsList.add("Blue");

      String colorsString = colorsList.toString();

      System.out.println("String: " + colorsString);
   }
}
```

Output:
String: [Red, Green, Blue]

- **ensureCapacity(minCapacity):**

- **What is the output of the following Java program fragment:**

```
import java.util.ArrayList;

public class Test {
   public static void main(String[] args) {
      ArrayList<Integer> numbers = new ArrayList<>(5);

      numbers.add(1);
      numbers.add(2);
      numbers.add(3);
      numbers.add(4);
      numbers.add(5);

      System.out.println("Current capacity: " +
numbers.size());

      numbers.ensureCapacity(10);

      numbers.add(6);
      numbers.add(7);
      numbers.add(8);
      numbers.add(9);
      numbers.add(10);

      System.out.println("Updated capacity: " +
numbers.size());
   }
}
```

Output:
Current capacity: 5
Updated capacity: 10

- **lastIndexOf(element):**

- **What is the output of the following Java program fragment:**

```
import java.util.ArrayList;
import java.util.Arrays;

public class Test {
   public static void main(String[] args) {
      ArrayList<String> fruits = new ArrayList<>
(Arrays.asList("Apple", "Banana", "Orange", "Apple",
"Mango"));

      int lastIndex = fruits.lastIndexOf("Apple");

      System.out.println("Last index of 'Apple': " +
lastIndex);

   }
}
```

Output:
Last index of 'Apple': 3

- **listIterator():**

- **What is the output of the following Java program fragment:**

```
import java.util.ArrayList;
import java.util.List;
import java.util.ListIterator;

public class Test {
   public static void main(String[] args) {
      List<String> fruits = new ArrayList<>();
      fruits.add("Apple");
      fruits.add("Banana");
      fruits.add("Orange");

      ListIterator<String> iterator = fruits.listIterator();

      System.out.println("Forward iteration:");
      while (iterator.hasNext()) {
         String fruit = iterator.next();
         System.out.println(fruit);
      }

      System.out.println("\nBackward iteration:");
      while (iterator.hasPrevious()) {
         String fruit = iterator.previous();
         System.out.println(fruit);
      }
   }
}
```

Output:

Forward iteration:
Apple
Banana
Orange

Backward iteration:
Orange
Banana
Apple

- **listIterator(index):**

- **What is the output of the following Java program fragment:**

```
import java.util.ArrayList;
import java.util.List;
import java.util.ListIterator;

public class Test {
   public static void main(String[] args) {
      List<String> fruits = new ArrayList<>();
      fruits.add("Apple");
      fruits.add("Banana");
      fruits.add("Orange");
      fruits.add("Mango");

      // Obtain a ListIterator starting from index 2
      ListIterator<String> iterator = fruits.listIterator(2);

      // Iterate over the elements from index 2 onwards
      System.out.println("Iteration from index 2
onwards:");
      while (iterator.hasNext()) {
         String fruit = iterator.next();
         System.out.println(fruit);
      }
   }
}
```

Output:

Iteration from index 2 onwards:
Orange
Mango

- **replaceAll(operator):**

- **What is the output of the following Java program fragment:**

```
import java.util.ArrayList;
import java.util.List;

public class Test {
   public static void main(String[] args) {
      List<String> fruits = new ArrayList<>();
      fruits.add("Apple");
      fruits.add("Banana");
      fruits.add("Orange");
      fruits.add("Mango");
      fruits.add("Apple");

      // Replace all occurrences of "Apple" with "Pineapple"
      fruits.replaceAll(fruit -> fruit.equals("Apple") ?
"Pineapple" : fruit);

      // Print the updated list
      System.out.println(fruits);
   }
}
```

Output:
[Pineapple, Banana, Orange, Mango, Pineapple]

- **sort(comparator):**

  - **What is the output of the following Java program fragment:**

```
import java.util.ArrayList;
import java.util.Comparator;
import java.util.List;

public class Test {
   public static void main(String[] args) {
      List<String> fruits = new ArrayList<>();
      fruits.add("Banana");
      fruits.add("Apple");
      fruits.add("Orange");
      fruits.add("Ant");
      fruits.add("Mango");

      // Sort the list in ascending order using a custom
comparator
      fruits.sort(Comparator.naturalOrder());

      // Print the sorted list
      System.out.println(fruits);
   }
}
```

Output:
[Ant, Apple, Banana, Mango, Orange]

- **trimToSize():**

- **What is the output of the following Java program fragment:**

```
import java.util.ArrayList;

public class Test {
   public static void main(String[] args) {
      // Create an ArrayList with an initial capacity of 10
      ArrayList<Integer> numbers = new ArrayList<>(10);

      numbers.add(1);
      numbers.add(2);
      numbers.add(3);
      numbers.add(4);
      numbers.add(5);

      // Trim the capacity of the ArrayList to match its size
      numbers.trimToSize();

      // Print the current capacity of the ArrayList
      System.out.println("Current capacity: " +
numbers.size());
   }
}
```

Output:

Current capacity: 5

- **What is the output of the following Java program fragment:**

```
package basicjava;
import java.util.ArrayList;
import java.util.Scanner;

public class JavaCode {
   public static void main(String[] args) {
      Scanner scan = new Scanner(System.in);
      ArrayList<Integer> number = new
ArrayList<Integer>();
      System.out.println("size = "+number.size());

      //adding elements
      number.add(10);
      number.add(20);
      number.add(30);
      number.add(3,40);
      System.out.println(number);
      System.out.println("size = "+number.size());
   }
}
```

Output:
size = 0
[10, 20, 30, 40]
size = 4

Here,
ArrayList = class, <Integer> = method
number.add(index: 3, element: 40);

or,

```
package basicjava;
import java.util.ArrayList;
import java.util.Scanner;

public class JavaCode {
   public static void main(String[] args) {
      Scanner scan = new Scanner(System.in);
      ArrayList<Integer> number = new
ArrayList<Integer>();
      System.out.println("size = "+number.size());

      //adding elements
      number.add(10);
      number.add(20);
      number.add(30);
      number.add(3,40);
      for (int i : number) {
         System.out.print(" "+i);
      }
      System.out.println();
      System.out.println("size = "+number.size());
   }
}
```

Output:

```
size = 0
 10 20 30 40
size = 4
```

or,

```
package basicjava;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.Scanner;

public class JavaCode {
   public static void main(String[] args) {
      Scanner scan = new Scanner(System.in);
      ArrayList<Integer> number = new
ArrayList<Integer>();
      System.out.println("size = "+number.size());

      //adding elements
      number.add(10);
      number.add(20);
      number.add(30);
      number.add(3,40);
      Iterator itr = number.iterator();
      while(itr.hasNext()){
         System.out.print(" "+itr.next());
      }
      System.out.println();
      System.out.println("size = "+number.size());
   }
}
```

Output:

```
size = 0
 10 20 30 40
size = 4
```

or,

```
import java.util.Scanner;
import java.util.ArrayList;
import java.util.Iterator;

public class main {
   public static void main(String[] args) {
      Scanner scan = new Scanner(System.in);
      ArrayList<Integer> number = new
ArrayList<Integer>();
      System.out.println("size = "+number.size());
      //adding elements
      number.add(10);
      number.add(20);
      number.add(30);
      number.add(3,40);
      Iterator a = number.iterator();
      for( ; a.hasNext() ; ){
         System.out.println(a.next());
      }
      System.out.println("size = "+number.size());
      scan.close();
   }
}
```

Output:

```
size = 0
10
20
30
40
size = 4
```

Here,
We have used Iterator class to print the arrays elements. Here iterator() is a method of Iterator itr = number.iterator() statement and itr is a iterator variable/object.
java.util.Iterator; is the Iterator class location. We must import this.

- **What is the output of the following Java program fragment:**

```
package basicjava;
import java.util.*;

public class JavaCode {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      System.out.print("How many numbers you want to
add: ");
      int n = sc.nextInt();
      ArrayList<Integer> num = new ArrayList<>();
      System.out.println("Enter array elements: ");
      for (int i = 0; i < n; i++) {
         num.add(sc.nextInt());
      }
      System.out.println("Array is: " + num);
      sc.close();
   }
}
```

Output:
How many numbers you want to add: 5
Enter array elements:
1 2 3 4 5
Array is: [1, 2, 3, 4, 5]

- **What is the output of the following Java program fragment:**

```
import java.util.Scanner;
import java.util.ArrayList;
import java.util.Iterator;

class arrayList{
   ArrayList<Integer> number;
   public arrayList(){
         number = new ArrayList<Integer>();
   }

   public void setNumber(){
      Scanner scan = new Scanner(System.in);
      for(int i=0; i<5; i++){
         number.add(scan.nextInt());
      }
      scan.close();
   }

   public void getNumber1(){
      for(int i=0; i<5; i++){
         System.out.println(number.get(i));
      }
   }

   public void getNumber2(){
      for(int a : number){
         System.out.println(a);
      }
   }
```

```
   public void getNumber3(){
      Iterator a = number.iterator();

      while(a.hasNext()){
         System.out.println(a.next());
      }
   }
}

public class Test{
   public static void main(String[] args) {
      arrayList ob = new arrayList();

      ob.setNumber();
      System.out.println();

      ob.getNumber1();
      System.out.println();
      ob.getNumber2();
      System.out.println();
      ob.getNumber3();
   }
}
```

Output:

1
2
3
4
5

1
2
3
4
5

1
2
3
4
5

1
2
3
4
5

- **What is the output of the following Java program fragment:**

```
package basicjava;
import java.util.ArrayList;
import java.util.Scanner;

public class JavaCode {
   public static void main(String[] args) {
      Scanner scan = new Scanner(System.in);
      ArrayList<Integer> number = new
ArrayList<Integer>();
      System.out.println("size = "+number.size());

      //adding elements
      number.add(10);
      number.add(20);
      number.add(30);
      number.add(3,40);
      System.out.println("ArrayList contains :
"+number);
      System.out.println();
      System.out.println("size = "+number.size());

      //Removing elements
      number.remove(2);
      System.out.println("After removing ArrayList
contains : "+number);
      number.removeAll(number);
      System.out.println("After removing all :
"+number);
   }
}
```

or,

```
package basicjava;
import java.util.ArrayList;
import java.util.Scanner;

public class JavaCode {
   public static void main(String[] args) {
      Scanner scan = new Scanner(System.in);
      ArrayList<Integer> number = new
ArrayList<Integer>();
      System.out.println("size = "+number.size());

      //adding elements
      number.add(10);
      number.add(20);
      number.add(30);
      number.add(3,40);
      System.out.println("ArrayList contains :
"+number);
      System.out.println();
      System.out.println("size = "+number.size());

      //Removing elements
      number.remove(2);
      System.out.println("After removing ArrayList
contains : "+number);
      number.clear();
      System.out.println("After removing all :
"+number);
   }
}
```

Output:
size = 0
ArrayList contains : [10, 20, 30, 40]

size = 4
After removing ArrayList contains : [10, 20, 40]
After removing all : [ ]

Here,
We have used remove method and clear method to remove arrays element.

- **What is the output of the following Java program fragment:**

```
package basicjava;
import java.util.*;

public class JavaCode {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      List<Integer> numbers = Arrays.asList(1, 2, 3, 4, 5, 6,
7, 8);
      for (int i = 0; i < numbers.size(); i++)
         System.out.print(numbers.get(i) + " ");
   }
}
```

Output:
1 2 3 4 5 6 7 8

- **What is the output of the following Java program fragment:**

```
package basicjava;
import java.util.ArrayList;
import java.util.Scanner;

public class JavaCode {
   public static void main(String[] args) {
      Scanner scan = new Scanner(System.in);
      ArrayList<Integer> number = new
ArrayList<Integer>();
      System.out.println("size = "+number.size());

      //adding elements
      number.add(10);
      number.add(20);
      number.add(30);
      number.add(3,40);
      System.out.println("ArrayList contains :
"+number);
      System.out.println();
      System.out.println("size = "+number.size());
      //checking that Arraylist empty or not
      boolean check = number.isEmpty();
      System.out.println("Arrylist empty : "+check);
   }
}
```

Output:
size = 0
ArrayList contains : [10, 20, 30, 40]

size = 4
Arrylist empty : false

Here,
We have used isEmpty method to check that the array's elements are empty or not.

- **What is the output of the following Java program fragment:**

```
package basicjava;
import java.util.ArrayList;
import java.util.Scanner;

public class JavaCode {
   public static void main(String[] args) {
      Scanner scan = new Scanner(System.in);
      ArrayList<Integer> number = new
ArrayList<Integer>();
      System.out.println("size = "+number.size());

      //adding elements
      number.add(10);
      number.add(20);
      number.add(30);
      number.add(3,40);
      System.out.println("ArrayList contains :
"+number);
```

```
        System.out.println();
        System.out.println("size = "+number.size());

        //checking that Arraylist empty or not
        number.clear();
        boolean check = number.isEmpty();
        System.out.println("Arrylist empty : "+check);
    }
}
```

Output:

```
size = 0
ArrayList contains : [10, 20, 30, 40]

size = 4
Arrylist empty : true
```

- **What is the output of the following Java program fragment:**

```
package basicjava;
import java.util.ArrayList;
import java.util.Scanner;

public class JavaCode {
   public static void main(String[] args) {
      Scanner scan = new Scanner(System.in);
      ArrayList<Integer> number = new
ArrayList<Integer>();
      System.out.println("size = "+number.size());

      //adding elements
      number.add(10);
      number.add(20);
      number.add(30);
      number.add(3,40);
      System.out.println("ArrayList contains :
"+number);
      System.out.println();
      System.out.println("size = "+number.size());

      //checking that Arraylist empty or not
      boolean check = number.isEmpty();
      System.out.println("Arrylist empty : "+check);
      boolean contain = number.contains(30);
      System.out.println("30 is in the list : "+contain);
      int pos = number.indexOf(40);
      System.out.println("The index of 40 is : "+pos);
```

```
        number.set(3,50);
        System.out.println("After setting : "+number);
        int x = number.get(0);
        System.out.println("index 0 = "+x);
    }
}
```

Output:

```
size = 0
ArrayList contains : [10, 20, 30, 40]

size = 4
Arrylist empty : false
30 is in the list : true
The index of 40 is : 3
After setting : [10, 20, 30, 50]
index 0 = 10
```

Here,
We have have used contains method to check any element is present in a array or not. We used indexOf() method to see the index of the element. We used set() method to replace already existing elements of an array. We used get() method to get any of element from the array index.

- **What is the output of the following Java program fragment:**

```
package basicjava;
import java.util.ArrayList;
import java.util.Scanner;

public class JavaCode {
   public static void main(String[] args) {
      Scanner scan = new Scanner(System.in);
      ArrayList<Integer> number1 = new
ArrayList<Integer>();
      ArrayList<Integer> number2 = new
ArrayList<Integer>();
      ArrayList<Integer> number3 = new
ArrayList<Integer>();

      number1.add(10);
      number1.add(20);
      number1.add(30);
      number1.add(40);
      System.out.println("Number1 = "+number1);

      number2.add(1);
      number2.add(2);
      number2.add(3);
      number2.add(4);
      System.out.println("Number2 = "+number2);

      number3.addAll(number1);
      System.out.println("Number3 = "+number3);
   }
}
```

or,

```
package basicjava;
import java.util.ArrayList;
import java.util.Scanner;

public class JavaCode {
   public static void main(String[] args) {
      Scanner scan = new Scanner(System.in);
      ArrayList<Integer> number1 = new
ArrayList<Integer>();
      ArrayList<Integer> number2 = new
ArrayList<Integer>();
      ArrayList<Integer> number3 = new
ArrayList<Integer>();
      int x;
      for (int i = 0; i < 4; i++) {
         x = scan.nextInt();
         number1.add(x);
      }

      System.out.println("Number1 = "+number1);

      number2.add(1);
      number2.add(2);
      number2.add(3);
      number2.add(4);
      System.out.println("Number2 = "+number2);

      number3.addAll(number1);
      System.out.println("Number3 = "+number3);
   }
}
```

Output:
Number1 = [10, 20, 30, 40]
Number2 = [1, 2, 3, 4]
Number3 = [10, 20, 30, 40]

Here,
We have used addAll( ) method to copy all the elements of number1 ArrayList and paste it in number3.

- **What is the output of the following Java program fragment:**

```
package basicjava;
import java.util.ArrayList;
import java.util.Scanner;

public class JavaCode {
   public static void main(String[] args) {
      Scanner scan = new Scanner(System.in);
      ArrayList<Integer> number1 , number2,
number3;
   number1 = new ArrayList<Integer>();
   number2 = new ArrayList<Integer>();
   number3 = new ArrayList<Integer>();
      int x;
      System.out.print("Enter 4 elements for
number1 : ");
      for (int i = 0; i < 4; i++) {
         x = scan.nextInt();
         number1.add(x);
```

```
    }

     System.out.println("number1 = "+number1);

     number2.add(1);
     number2.add(2);
     number2.add(3);
     number2.add(4);
     System.out.println("number2 = "+number2);

     number3.addAll(number1);
     System.out.println("number3 = "+number3);
     boolean result = number1.equals(number2);
     System.out.println("number1 == number2 :
"+result);
   }
}
```

Output:

```
Enter 4 elements for number1 : 1 2 3 4
number1 = [1, 2, 3, 4]
number2 = [1, 2, 3, 4]
number3 = [1, 2, 3, 4]
number1 == number2 : true
```

Here,
We used equals( ) method to see that number1 arraylist and number2 arralist are equal or not.

or,

```
package basicjava;
import java.util.ArrayList;
import java.util.Scanner;

public class JavaCode {
   public static void main(String[] args) {
       Scanner scan = new Scanner(System.in);
       ArrayList<Integer> number1 = new
ArrayList<Integer>();
       ArrayList<Integer> number2 = new
ArrayList<Integer>();
       ArrayList<Integer> number3 = new
ArrayList<Integer>();
       System.out.print("Enter 4 elements for
number1 : ");
       for (int i = 0; i < 4; i++) {
        number1.add(scan.nextInt());
       }

       System.out.println("number1 = "+number1);

       number2.add(1);
       number2.add(2);
       number2.add(3);
       number2.add(4);
       System.out.println("number2 = "+number2);

       number3.addAll(number1);
```

```
        System.out.println("number3 = "+number3);
        boolean result = number1.equals(number2);
        System.out.println("number1 == number2 :
"+result);
        result = number1.equals(number3);
        System.out.println("number1 == number3 :
"+result);
    }
}
```

Output:

```
Enter 4 elements for number1 : 10 20 30 40
number1 = [10, 20, 30, 40]
number2 = [1, 2, 3, 4]
number3 = [10, 20, 30, 40]
number1 == number2 : false
number1 == number3 : true
```

- **What is the output of the following Java program fragment:**

```
package basicjava;
import java.util.ArrayList;
import java.util.Collections;
import java.util.Scanner;

public class JavaCode {
   public static void main(String[] args) {
      Scanner scan = new Scanner(System.in);
      ArrayList<Integer> number = new ArrayList<>();

      System.out.print("How many number do you
want to enter : ");
      int n = scan.nextInt();

      System.out.print("Enter "+n+" numbers : ");
      for (int i = 0; i < n; i++) {

         number.add(number1.add(scan.nextInt());
);
      }
      System.out.println("Before sorting : "+number);
      Collections.sort(number);
      System.out.println("After sorting in ascending :
"+number);

Collections.sort(number,Collections.reverseOrder());
      System.out.println("After sorting in descending :
"+number);
   }
}
```

Output:
How many number do you want to enter : 6
Enter 6 numbers : 20 -3 18 92 0 2
Before sorting : [20, -3, 18, 92, 0, 2]
After sorting in ascending : [-3, 0, 2, 18, 20, 92]
After sorting in descending : [92, 20, 18, 2, 0, -3]

Here,
We have used sort method of Collections class to sort the ArrayList elements. After then we have used reverseOrder( ) method of Collections class to reverse the sorted elements of number ArrayList.

- **What is the output of the following Java program fragment:**

```
package basicjava;
import java.util.ArrayList;
import java.util.Collections;
import java.util.Scanner;

public class JavaCode {
   public static void main(String[] args) {
      Scanner scan = new Scanner(System.in);
      ArrayList<Integer> number = new ArrayList<>
();

      System.out.print("How many number do you
want to enter : ");
      int n = scan.nextInt();
      int x;
      System.out.print("Enter "+n+" numbers : ");
```

```
      for (int i = 0; i < n; i++) {
          x = scan.nextInt();
          number.add(x);
       }
       System.out.println("Before sorting :
"+number);

Collections.sort(number,Collections.reverseOrder())
;
       System.out.println("After sorting in descending
: "+number);
   }
}
```

Output:

```
How many number do you want to enter : 6
Enter 6 numbers : 20 -3 18 92 0 2
Before sorting : [20, -3, 18, 92, 0, 2]
After sorting in descending : [92, 20, 18, 2, 0, -3]
```

- **Multidimensional ArrayList :**

In Java, a multidimensional ArrayList is a data structure that allows you to create a list of lists, effectively forming a 2D or higher-dimensional data structure. Unlike arrays, which have a fixed size, ArrayLists can dynamically resize, making them more flexible for handling collections of data.

To work with a multidimensional ArrayList, you need to create an ArrayList of ArrayLists (or an ArrayList of any other type of ArrayList, depending on the dimensionality you want to achieve).

- **What is the output of the following Java program fragment:**

```
import java.util.ArrayList;

public class MultiDimensionalArrayListExample {
    public static void main(String[] args) {
        // Creating a 3x3 2D ArrayList
        ArrayList<ArrayList<Integer>> matrix = new
ArrayList<>();

        // Number of rows and columns
        int rows = 3;
        int cols = 3;

        // Loop to create rows and populate the 2D ArrayList
        for (int i = 0; i < rows; i++) {
            ArrayList<Integer> row = new ArrayList<>();
            for (int j = 0; j < cols; j++) {
                row.add(i * cols + j + 1); // Adding elements (1 to
9 in this example)
            }
            matrix.add(row); // Adding the row to the 2D
ArrayList
        }

        // Accessing elements and printing the 2D ArrayList
        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                System.out.print(matrix.get(i).get(j) + " ");
            }
```

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

Output:

1 2 3
4 5 6
7 8 9

- **Create multiple objects in java using ArrayList:**

We can create multiple object using ArrayList.

- **What is the output of the following Java program fragment:**

```
import java.util.ArrayList;

class Person {
  private String name;

  public Person(String name) {
    this.name = name;
  }

  public String getName() {
    return name;
  }
}

public class MultipleObjectCreationExample {
  public static void main(String[] args) {
    ArrayList<Person> personList = new ArrayList<>();

    // Create multiple Person objects and add them to the
ArrayList

    personList.add(new Person("John"));
    personList.add(new Person("Jane"));
    personList.add(new Person("Alice"));

    System.out.println("Person name:
"+personList.get(0).getName());
     System.out.println();
    // Perform operations using the person objects in the
ArrayList
```

```
    for (Person person : personList) {
      // Access each person object

      System.out.println("Person name: " +
person.getName());
    }
  }
}
```

Output:
Person name: John

Person name: John
Person name: Jane
Person name: Alice

- **What is the output of the following Java program fragment:**

```
package FirstPackage;

import java.util.Scanner;
import java.util.ArrayList;
import java.util.List;

class Student{
    String name;
    int roll;
    int marks;
```

```
    public Student(String n, int r, int m){
        this.name = n;
        this.roll = r;
        this.marks = m;
    }

    public void display(){
        System.out.println("Name: "+this.name);
        System.out.println("Roll: "+this.roll);
        System.out.println("Mark: "+this.marks);
    }
}

public class multipleObject {
    public static void main(String[] args) {

        List<Student> studentList = new ArrayList<Student>();

        String ns[] = {"Hamim", "Hridi", "Jim"};
        int rs[] = {13, 12, 10};
        int ms[] = {90, 89,85};

        for(int i=0; i<3; i++){
            studentList.add(new Student(ns[i],rs[i],ms[i]));
        }

        studentList.get(1).display();

        System.out.println();

        for(int i=0; i<3; i++){
```

```
            studentList.get(i).display();
            System.out.println();
        }
    }
}
```

Output:

Name: Hridi
Roll: 12
Mark: 89

Name: Hamim
Roll: 13
Mark: 90

Name: Hridi
Roll: 12
Mark: 89

Name: Jim
Roll: 10
Mark: 85

- **What is the output of the following Java program fragment:**

```
package FirstPackage;

import java.util.Scanner;
import java.util.ArrayList;

class Student{
   String name;
   int roll;
   int marks;

 public Student(String n, int r, int m){
 this.name = n;
 this.roll = r;
 this.marks = m;
 }

 public void display(){
 System.out.println("Name: "+this.name);
 System.out.println("Roll: "+this.roll);
 System.out.println("Mark: "+this.marks);
 }
}
```

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

        ArrayList<Student> studentList = new
ArrayList<Student>();

        String ns[] = {"Hamim", "Hridi", "Jim"};
        int rs[] = {13, 12, 10};
        int ms[] = {90, 89,85};

        for(int i=0; i<3; i++){
            Student s = new Student(ns[i], rs[i], ms[i]);
            studentList.add(s);
        }


        studentList.get(1).display();
        System.out.println();

       for(int i=0; i<3; i++){
           studentList.get(i).display();
           System.out.println();
       }
    }
}
```

Output:
Name: Hridi
Roll: 12
Mark: 89

Name: Hamim
Roll: 13
Mark: 90

Name: Hridi
Roll: 12
Mark: 89

Name: Jim
Roll: 10
Mark: 85

- **What is the output of the following Java program fragment:**

```
package FirstPackage;

import java.util.Scanner;
import java.util.ArrayList;
import java.util.List;

class Student{
   String name;
   int roll;
   int marks;

   public Student(String n, int r, int m){
      this.name = n;
      this.roll = r;
      this.marks = m;
   }

   public void display(){
      System.out.println("Name: "+this.name);
      System.out.println("Roll: "+this.roll);
      System.out.println("Mark: "+this.marks);
   }
}

public class multipleObject {
   public static void main(String[] args) {

      List<Student> studentList = new
ArrayList<Student>();
```

```
        String ns[] = {"Hamim", "Hridi", "Jim"};
        int rs[] = {13, 12, 10};
        int ms[] = {90, 89,85};

        for(int i=0; i<3; i++){
            Student s = new Student(ns[i], rs[i], ms[i]);
            studentList.add(s);
        }

        studentList.get(1).display();

        System.out.println();

        for(int i=0; i<3; i++){
            studentList.get(i).display();
            System.out.println();
        }
    }
}
```

Output:
Name: Hridi
Roll: 12
Mark: 89

Name: Hamim
Roll: 13
Mark: 90

Name: Hridi
Roll: 12
Mark: 89

Name: Jim
Roll: 10
Mark: 85

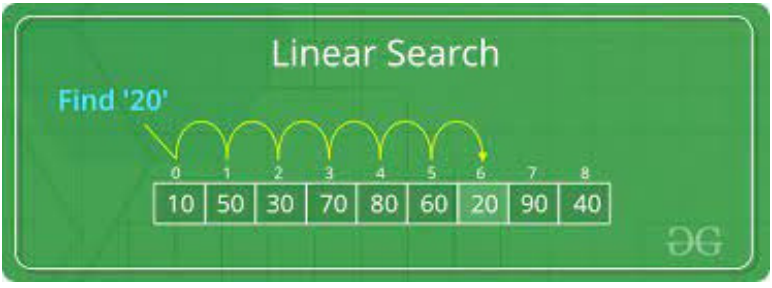
4. Searching

- **Searching :**

1. Linear Search
2. Binary Search
3. Modified Binary Search
4. Binary Search on 2D Arrays

1. Linear Search :

Linear search is a simple searching algorithm that iterates through an array or a list to find a specific element. It sequentially checks each element until the target element is found or until the end of the list is reached.



Time Complexity:
Best case: O(1)
Worst case: O(n)

Where n is the size of the array.

Space Complexity:
O(1)

- **What is the output of the following Java program fragment:**

```
public class Hamim {
 public static int linearSearch(int[] arr, int target) {
  for (int i = 0; i < arr.length; i++) {
   if (arr[i] == target) {
    return i;
   }
  }
  return -1;
 }

public static void main(String[] args) {
 int[] numbers = { 12, 45, 67, 23, 9, 8, 34, 78 };
 int targetElement = 23;
 int index = linearSearch(numbers, targetElement);

 if (index != -1) {
 System.out.println("Element " + targetElement + "
found at index: " + index);
 }

 else {
 System.out.println("Element " + targetElement + "
not found in the array.");
 }
 }
}
```

Output:

Element 23 found at index: 3

- **What is the output of the following Java program fragment:**

```
public class Hamim {
   public static int linearSearch(int[] arr, int target) {
      for (int num : arr) {
         if (num == target) {
            return num;
         }
      }
      return Integer.MAX_VALUE ;
   }

   public static void main(String[] args) {
      int[] numbers = {12, 45, 67, 23, 9, 8, 34, 78};
      int targetElement = 230;

      int element = linearSearch(numbers,
targetElement);

      if (element != Integer.MAX_VALUE) {
         System.out.println("Element " + element + "
found");
      } else {
         System.out.println("Element " + targetElement + "
not found in the array.");
      }
   }
}
```

Output:
Element 230 not found in the array.

- **What is the output of the following Java program fragment:**

```
import java.util.Scanner;

public class Test {

   public static int linearSearch(int arr[], int target) {
      for (int index = 0; index < arr.length; index++) {
         if (target == arr[index]) {
            return index;
         }
      }
      return -1;

   }

   public static void main(String[] args) {
      Scanner scan = new Scanner(System.in);
      int n;
 System.out.print("How many number do you want
to enter : ");
 n = scan.nextInt();
 int array[] = new int[n];

 for (int i = 0; i < n; i++) {
 System.out.print("Enter array[" + i + "] = ");
 array[i] = scan.nextInt();
 }
```

```
        System.out.print("Enter checking item : ");
        int targetNumber = scan.nextInt();
        int x = linearSearch(array, targetNumber);

        if (x != -1) {
            System.out.println("The number " +
targetNumber + "found at " + x);
        } else {
            System.out.println("Number doesn't found
!");
        }
        scan.close();
    }
}
```

Output:

How many number do you want to enter : 5
Enter array[0] = 57
Enter array[1] = 9
Enter array[2] = 3
Enter array[3] = 88
Enter array[4] = 0
Enter checking item : 88
The number 88found at 3

- **What is the output of the following Java program fragment:**

```
import java.util.Scanner;

public class Test {

   public static int linearSearch(int arr[], int length, int
target) {
      for (int index = 0; index < length; index++) {
         if (target == arr[index]) {
            return index;
         }
      }
      return -1;

   }

   public static void main(String[] args) {
      Scanner scan = new Scanner(System.in);
      System.out.print("How many number do you
want to enter : ");
      int length = scan.nextInt();
      int array[] = new int[length];

      for (int i = 0; i < length ; i++) {
         System.out.print("Enter array[" + i + "] = ");
         array[i] = scan.nextInt();
      }
```

```
        System.out.print("Enter checking item : ");
        int targetNumber = scan.nextInt();
        int x = linearSearch(array, length, targetNumber);

        if (x != -1) {
            System.out.println("The number " +
targetNumber + " found at " + x);
        } else {
            System.out.println("Number doesn't found !");
        }
        scan.close();
    }
}
```

Output:

```
How many number do you want to enter : 5
Enter array[0] = 2
Enter array[1] = 5
Enter array[2] = 8
Enter array[3] = 90
Enter array[4] = 3
Enter checking item : 90
The number 90 found at 3
```

- **What is the output of the following Java program fragment:**

```
public class Hamim {

    public static void main(String[] args) {
        String name = "Hridi";
        char target = 'r';

        boolean bol = linearSearch(name, target);

        if (bol) {
            System.out.println(target + " match");
        } else {
            System.out.println(target+ " not match");
        }
    }

    static boolean linearSearch(String str, char target) {
        for (int i=0; i<str.length(); i++ ) {
            if (str.charAt(i) == target) {
                return true;
            }
        }
        return false;
    }

}
```

Output:
r match

- **What is the output of the following Java program fragment:**

```
public class Hamim {

    public static void main(String[] args) {
        String name = "Hridi";
        char target = 'k';

        boolean bol = linearSearch(name, target);

        if (bol) {
            System.out.println(target + " match");
        } else {
            System.out.println(target+ " not match");
        }
    }

    static boolean linearSearch(String str, char target) {
        for (char ch : str.toCharArray() ) {
            if (ch == target) {
                return true;
            }
        }
        return false;
    }

}
```

Output:
k not match

- **What is the output of the following Java program fragment:**

```
import java.util.Arrays;
public class Hamim {
   public static void main(String[] args) {
      String name = "Hridi";
      char target = 'k';

System.out.println(Arrays.toString(name.toCharArray()));
      boolean bol = linearSearch(name, target);

      if (bol) {
         System.out.println(target + " match");
      } else {
         System.out.println(target+ " not match");
      }
   }

   static boolean linearSearch(String str, char target) {
      for (char ch : str.toCharArray() ) {
         if (ch == target) {
            return true;
         }
      }
      return false;
   }
}
```

Output:

```
[H, r, i, d, i]
k not match
```

- **What is the output of the following Java program fragment:**

```
public class Hamim {
   public static int minimumNumber(int[] arr) {
      int temp = arr[0];
      for (int i = 1; i < arr.length; i++) {
         if (temp > arr[i]) {
            temp = arr[i];
         }
      }
      return temp;
   }

   public static void main(String[] args) {
      int[] numbers = { 12, 45, 67, 23, 9, 8, 34, 78 };

      System.out.println("Minimum number =
"+minimumNumber(numbers));
   }
}
```

Output:
Minimum number = 8

- **What is the output of the following Java program fragment:**

```
public class Hamim {
   public static int maximumNumber(int[] arr) {
      int temp = arr[0];
      for (int i = 1; i < arr.length; i++) {
         if (temp < arr[i]) {
            temp = arr[i];
         }
      }
      return temp;
   }

   public static void main(String[] args) {
      int[] numbers = { 12, 45, 67, 23, 9, 8, 34, 78 };

      System.out.println("Maximum number =
"+maximumNumber(numbers));
   }
}
```

Output:
Maximum number = 78

- **What is the output of the following Java program fragment:**

```
public class Hamim {

    public static int[] linearSearch2DArray(int[][] arr, int
target) {
        for (int row = 0; row < arr.length; row++) {
            for (int col = 0; col < arr[row].length; col++) {
                if (arr[row][col] == target) {
                    return new int[]{row, col};
                }
            }
        }
        return new int[]{-1, -1};
    }

    public static void main(String[] args) {
        int[][] matrix = {
            {1, 2, 3},
            {4, 5, 6},
            {7, 8, 9}
        };
        int targetElement = 5;

        int[] result = linearSearch2DArray(matrix,
targetElement);

        if (result[0] != -1 && result[1] != -1) {
            System.out.println("Element " + targetElement + "
found at row: " + result[0] + ", col: " + result[1]);
```

```
        } else {
            System.out.println("Element " +
targetElement + " not found in the 2D array.");
        }
    }
}
```

Output:
Element 5 found at row: 1, col: 1

- **What is the output of the following Java program fragment:**

```
import java.util.Arrays;

public class Hamim {

    public static int[] linearSearch2DArray(int[][] arr, int
target) {
        for (int row = 0; row < arr.length; row++) {
            for (int col = 0; col < arr[row].length; col++) {
                if (arr[row][col] == target) {
                    return new int[]{row, col};
                }
            }
        }
        return new int[]{-1, -1};
    }
```

```
    public static void main(String[] args) {
        int[][] matrix = {
            {23, 4, 1},
            {18, 12, 3, 9},
            {78, 99, 34, 56},
            {18, 12}
        };
        int targetElement = 34;

        int[] result = linearSearch2DArray(matrix,
targetElement);

        System.out.println(Arrays.toString(result));
    }
}
```

Output:

[2, 2]

- **What is the output of the following Java program fragment:**

```
public class Hamim {

   public static int MaxValue(int[][] arr) {
      int max = Integer.MIN_VALUE;
      for (int row = 0; row < arr.length; row++) {
         for (int col = 0; col < arr[row].length; col++) {
            if (arr[row][col] > max) {
               max = arr[row][col];
            }
         }
      }
      return max;
   }

   public static void main(String[] args) {
      int[][] matrix = {
            { 23, 4, 1 },
            { 18, 12, 3, 9 },
            { 78, 99, 34, 56 },
            { 18, 12 }
      };

      int result = MaxValue(matrix);

      System.out.println(result);
   }
}
```

Output:

99

- **What is the output of the following Java program fragment:**

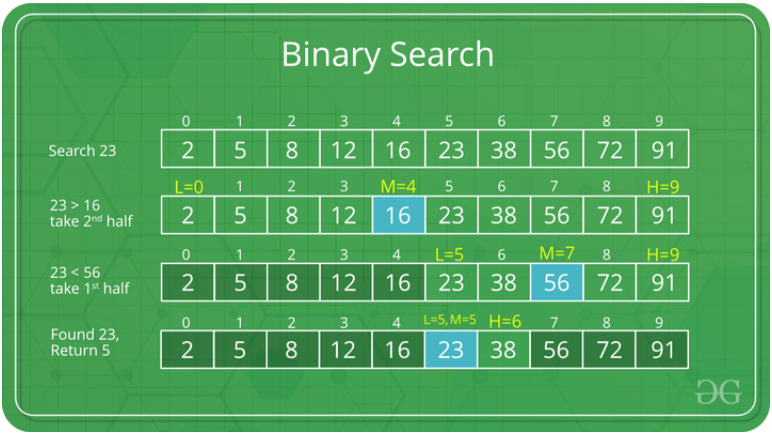
```
public class Hamim {

  public static int MinValue(int[][] arr) {
    int min = Integer.MAX_VALUE;
    for (int row = 0; row < arr.length; row++) {
      for (int col = 0; col < arr[row].length; col++) {
        if (arr[row][col] < min) {
          min = arr[row][col];
        }
      }
    }
    return min;
  }

  public static void main(String[] args) {
    int[][] matrix = {
        { 23, 4, 1 },
        { 18, 12, 3, 9 },
        { 78, 99, 34, 56 },
        { 18, 12 }
    };

    int result = MinValue(matrix);

    System.out.println(result);
  }
}
```

Output:

1

2. Binary Search :

Binary search is a search algorithm that finds the position of a target value within a sorted array. It works by repeatedly dividing the search space in half until the target value is found or it is determined that the target value does not exist in the array.



Time Complexity:
Best case: O(1)
Worst case: O(log n)

Where n is the size of the array.

Space Complexity:
O(1)

- **What is the output of the following Java program fragment:**

```
public class Test {
  public static int binarySearch(int[] arr, int target) {
    int left = 0;
    int right = arr.length - 1;

    while (left <= right) {
      int mid = left + (right - left) / 2;

      if (arr[mid] == target) {
        return mid;
      }

      if (arr[mid] > target) {
        right = mid - 1;
      }

      else {
        left = mid + 1;
      }
    }

    return -1;
  }

  public static void main(String[] args) {
    int[] sortedArray = { 1, 3, 5, 7, 9, 11, 13, 15, 17 };
    int target = 11;
```

```
    int index = binarySearch(sortedArray, target);

    if (index != -1) {
      System.out.println("Target found at index: " + index);
    } else {
      System.out.println("Target not found in the array.");
    }
  }
}
```

Output:
Target found at index: 5

- **Ascending order:**

- **What is the output of the following Java program fragment:**

```
public class Test {
   public static int binarySearch(int[] arr, int target) {
      int left = 0;
      int right = arr.length - 1;

      while (left <= right) {
         int mid = left + (right - left) / 2;



         if (target < arr[mid]) {
            right = mid - 1;
         }
         else if (target > arr[mid]) {
            left = mid + 1;
         }
         else{
            return mid;
         }

      }

      return -1;
   }
```

```
    public static void main(String[] args) {
        int[] sortedArray = {1, 3, 5, 7, 9, 11, 13, 15, 17};
        int target = 11;
        int index = binarySearch(sortedArray, target);

        if (index != -1) {
            System.out.println("Target found at index: "
+ index);
        } else {
            System.out.println("Target not found in the
array.");
        }
    }
}
```

Output:

Target found at index: 5

- **Descending order:**

- **What is the output of the following Java program fragment:**

```
public class Test {
  public static int binarySearch(int[] arr, int target) {
    int left = 0;
    int right = arr.length - 1;

    while (left <= right) {
      int mid = left + (right - left) / 2;



      if (target < arr[mid]) {
        left = mid + 1;
      }
      else if (target > arr[mid]) {
        right = mid - 1;
      }
      else{
        return mid;
      }

    }

    return -1;
  }
```

```
  public static void main(String[] args) {
    int[] sortedArray = {17, 15, 13, 11, 9, 7, 5, 3, 1};
    int target = 11;
    int index = binarySearch(sortedArray, target);

    if (index != -1) {
      System.out.println("Target found at index: " + index);
    } else {
      System.out.println("Target not found in the array.");
    }
  }
}
```

Output:

Target found at index: 3

- **When we don't know ascending or descending order:**

- **What is the output of the following Java program fragment:**

```
public class Test {
   public static int binarySearch(int[] arr, int target) {
      int left = 0;
      int right = arr.length - 1;

      boolean isAsc = arr[left] < arr[right];

      while (left <= right) {
         int mid = left + (right - left) / 2;

         if (arr[mid] == target) {
            return mid;
         }

         if (isAsc) {
            if (arr[mid] > target) {
               right = mid - 1;
            } else {
               left = mid + 1;
            }
         }
         else {
            if (arr[mid] < target) {
               right = mid - 1;
```

```
            } else {
                left = mid + 1;
            }
        }

    }

    return -1;
  }

  public static void main(String[] args) {
    int[] sortedArray = { 1, 3, 5, 7, 9, 11, 13, 15, 17 };
    int target = 11;
    int index = binarySearch(sortedArray, target);

    if (index != -1) {
        System.out.println("Target found at index: "
+ index);
    } else {
        System.out.println("Target not found in the
array.");
    }
  }
}
```

Output:

Target found at index: 5

- **Binary search(First occurrence) from sorted array.**

```
public class Minian {
  static int binarySearch(int arr[], int target) {
    int left = 0;
    int right = arr.length - 1;

    while(left<=right){
      int mid = left + (right-left)/2;

      if(arr[mid] == target){
        if(mid == 0 || arr[mid-1] < target){
          return mid;
        }
        else{
          right = mid-1;
        }

      }else if(arr[mid]<target){
        left = mid+1;
      }else{
        right = mid-1;
      }
    }

    return -1;
  }

  public static void main(String args[]) {
    int arr[] = { 1, 2, 2, 2, 3, 4, 4, 5, 6 };
```

```
    int target = 4;
    int n = arr.length;

    int result = binarySearch(arr, target);
    if (result != -1) {
      System.out.println("First occurrence of " +
target + " is at index " + result);
    } else {
      System.out.println(target + " not found in the
array");
    }
  }
}
```

Output:

First occurrence of 4 is at index 5

- **Binary search(Last occurrence) from sorted array.**

```
public class Minian {
  static int binarySearch(int arr[], int target) {
    int left = 0;
    int right = arr.length - 1;

    while(left<=right){
      int mid = left + (right-left)/2;

      if(arr[mid] == target){
        if(mid == arr.length - 1 || arr[mid+1] > target){
          return mid;
        }
        else{
          right = mid+1;
        }

      }else if(arr[mid]<target){
        left = mid+1;
      }else{
        right = mid-1;
      }
    }

    return -1;
  }

  public static void main(String args[]) {
    int arr[] = { 1, 2, 2, 2, 3, 4, 4, 5, 6 };
```

```
 int target = 4;
 int n = arr.length;

 int result = binarySearch(arr, target);
 if (result != -1) {
 System.out.println("First occurrence of " + target +
" is at index " + result);
 } else {
 System.out.println(target + " not found in the
array");
 }
 }
}
```

Output:

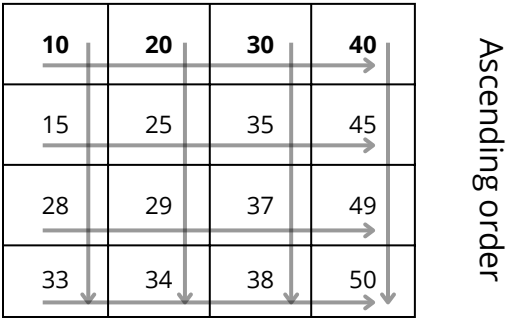
Last occurrence of 4 is at index 6

- **Binary Search in 2D Arrays:**

Type 1 : Unsorted 2D

Rows and columns are in ascending order.

Ascending order



Target : 37

- **What is the output of the following Java program fragment:**

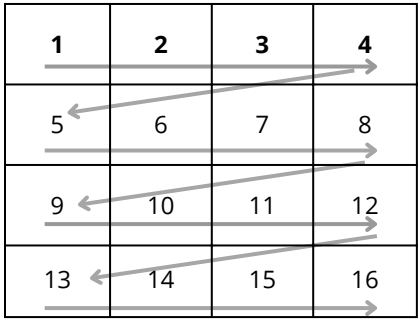
```
import java.util.Arrays;

public class Test {
   public static int[] binarySearchIn2DArray(int[][]
matrix, int target) {
       int r = 0;
       int c = matrix[0].length - 1;


       while (r < matrix.length && c>=0) {
          if(matrix[r][c] == target){
             return new int[] {r, c};
          }
          if(matrix[r][c] < target){
             r++;
          } else{
             c--;
          }

       }

       return new int[]{-1,-1};
   }
```

```
    public static void main(String[] args) {
        int[][] arr = {
            {10, 20, 30, 40},
            {15, 25, 35, 45},
            {28, 29, 37, 49},
            {33, 34, 38, 50}
        } ;


System.out.println(Arrays.toString(binarySearchIn2
DArray(arr, 37)));
    }
}
```

Output:

[2, 2]

Type 2 : Sorted 2D



Target : 3

- **What is the output of the following Java program fragment:**

```
import java.util.Arrays;

public class Test {

   // search in the row provided between the cols provided
   static int[] binarySearch(int[][] matrix, int row, int cStart, int cEnd, int target){
       while(cStart <= cEnd){
          int mid = cStart + (cEnd - cStart) / 2;
          if(matrix[row][mid] == target){
             return new int[]{row, mid};
          }
```

```
            if(matrix[row][mid] < target){
                cStart = mid + 1;
            } else {
                cEnd = mid - 1;
            }

        }
        return new int[]{-1, -1};
    }

    public static int[] binarySearchIn2DArray(int[][] matrix,
int target) {
        int rows = matrix.length;
        int cols = matrix[0].length ;


        if(rows == 1){
            return binarySearch(matrix, 0, 0, cols-1, target);
        }

        int rStart = 0;
        int rEnd = rows - 1;
        int cMid = cols / 2;

        // run the loop till 2 rows are remaining
        while(rStart < (rEnd -1)){ // while this is true it will
have more than 2 rows
            int mid = rStart + (rEnd - rStart) / 2;

            if(matrix[mid][cMid] == target){
                return new int[] {mid, cMid};
            }
```

```
            if(matrix[mid][cMid] < target){
                rStart = mid +1;
            } else{
                rEnd = mid -1;
            }
        }

        // now we have two rows
        // check wheather the target is in the col of 2 rows

        if(matrix[rStart][cMid] == target){
            return new int [] {rStart, cMid};
        }
        if(matrix[rStart + 1][cMid] == target){
            return new int [] {rStart +1, cMid};
        }

        // search in 1st half
        if(target <= matrix[rStart][cMid - 1]){
            return binarySearch(matrix, rStart, 0, cMid - 1,
target);
        }
        // search in 2nd half
        if(target >= matrix[rStart][cMid + 1]){
            return binarySearch(matrix, rStart, cMid + 1, cols
- 1, target);
        }
        // search in 3rd half
        if(target <= matrix[rStart + 1][cMid - 1]){
            return binarySearch(matrix, rStart + 1, 0, cMid - 1,
target);
        }
```

```
        // search in 4th half
        if(target <= matrix[rStart + 1][cMid + 1]){
            return binarySearch(matrix, rStart + 1, cMid + 1,
cols - 1, target);
        }

        return new int[]{-1,-1};
    }

    public static void main(String[] args) {
        int[][] arr = {
            {  1,  2,  3,  4},
            {  5,  6,  7,  8},
            {  9, 10, 11, 12},
            { 13, 14, 15, 16}
        } ;


System.out.println(Arrays.toString(binarySearchIn2DA
rray(arr, 3)));
    }
}
```

Output:

[0, 2]

Type 3 : Assuming the 2D Array as 1D

2	**4**	**6**	**8**
10	12	14	16
18	20	22	24
26	28	30	32

Target : 20

- **What is the output of the following Java program fragment:**

```
public class Hamim {
   public static int[] binarySearchIn2DArray(int[][] matrix,
int target) {
       int rows = matrix.length;
       int cols = matrix[0].length;
       int left = 0;
       int right = rows * cols - 1;

       while (left <= right) {
          int mid = left + (right - left) / 2;
          int midValue = matrix[mid / cols][mid % cols];
```

```
            if (midValue == target) {
                return new int[]{mid / cols, mid % cols};
            } else if (midValue < target) {
                left = mid + 1;
            } else {
                right = mid - 1;
            }
        }

        return new int[]{-1, -1};
    }

    public static void main(String[] args) {
        int[][] sortedMatrix = {
            {2, 4, 6, 8},
            {10, 12, 14, 16},
            {18, 20, 22, 24},
            {26, 28, 30, 32}
        };
        int target = 20;

        int[] result = binarySearchIn2DArray(sortedMatrix,
target);
        int row = result[0];
        int col = result[1];

        if (row != -1 && col != -1) {
            System.out.println("Target found at row " + row
+ " and column " + col);
        } else {
```

```
            System.out.println("Target not found in the 2D
array.");
        }
    }
}
```

Output:

Target found at row 2 and column 1

5. Sorting

1. Bubble Sort
2. Selection Sort
3. Insertion Sort
4. Cycle Sort

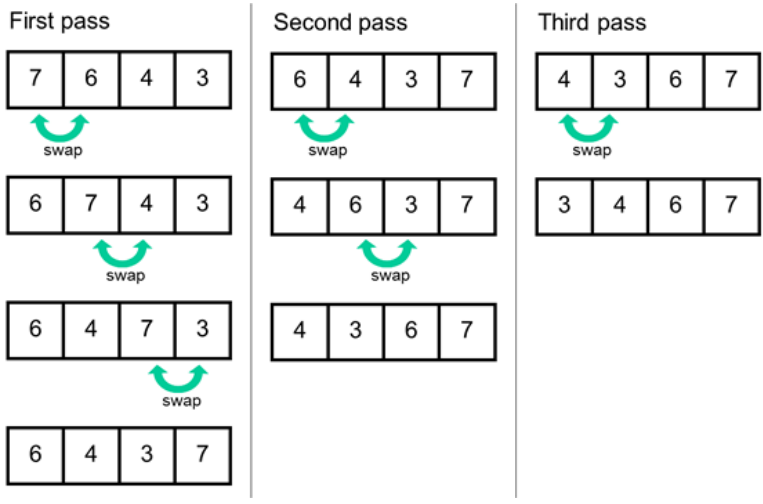
- **In built Java Sorting :**

```
import java.util.Arrays;

public class Test {
    public static void main(String[] args) {
        int[] numbers = {5, 2, 8, 1, 6};

        // Sorting in ascending order
        Arrays.sort(numbers);

        System.out.println("Sorted Array (Ascending
Order): " + Arrays.toString(numbers));
    }
}
```

Output:
Sorted Array (Ascending Order): [1, 2, 5, 6, 8]

1. Bubble Sort :

Bubble sort is a simple sorting algorithm that repeatedly steps through the list to be sorted, compares adjacent elements, and swaps them if they are in the wrong order. The process is repeated until the entire list is sorted.



Time Complexity:
Best case: O(n)
Worst case: O(n^2)

Where n is the size of the array.

Space Complexity:
O(1)

- **What is the output of the following Java program fragment:**

```
import java.util.Arrays;

public class Hridi {
  public static void bubbleSort(int[] arr) {
    int n = arr.length;
    boolean swapped;

    for (int i = 0; i < n - 1; i++) {
      swapped = false;

      for (int j = 0; j < n - 1 - i; j++) {
        if (arr[j] > arr[j + 1]) {
          // Swap arr[j] and arr[j+1]
          int temp = arr[j];
          arr[j] = arr[j + 1];
          arr[j + 1] = temp;
          swapped = true;
        }
      }
      // If no two elements were swapped in the inner
loop, the array is already
      // sorted
      if (!swapped) {
        break;
      }
    }
  }
```

```
 public static void main(String[] args) {
 int[] arr = { 64, 34, 25, 12, 22, 11, 90 };
 System.out.println("Original array: " +
Arrays.toString(arr));

 bubbleSort(arr);

 System.out.println("Sorted array: " +
Arrays.toString(arr));
 }
}
```

Output:

Original array: [64, 34, 25, 12, 22, 11, 90]
Sorted array: [11, 12, 22, 25, 34, 64, 90]

- **What is the output of the following Java program fragment:**

```
import java.util.Arrays;

public class Hridi {
    public static void bubbleSort(int[] arr) {
        int n = arr.length;
        boolean swapped;

        for (int i = 0; i < n - 1; i++) {
            swapped = false;

            for (int j = 0; j < n - i - 1; j++) {
                if (arr[j] < arr[j + 1]) {
                    // Swap arr[j] and arr[j+1]
                    int temp = arr[j];
                    arr[j] = arr[j + 1];
                    arr[j + 1] = temp;
                    swapped = true;
                }
            }
            // If no two elements were swapped in the inner loop, the array is already sorted
            if (!swapped) {
                break;
            }
        }
    }
```

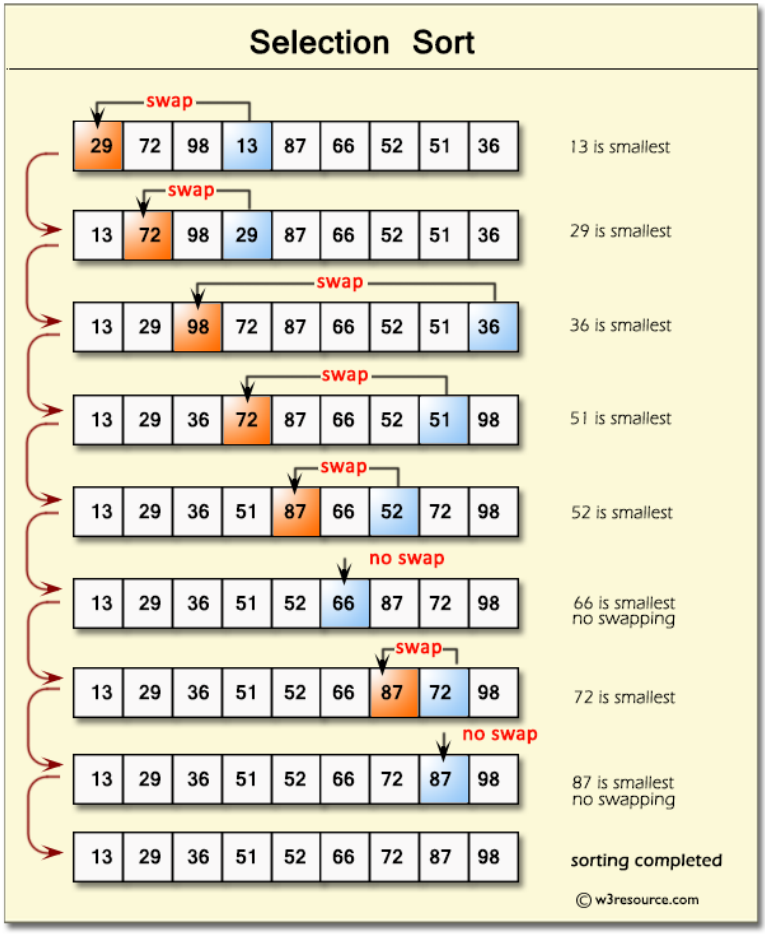
```
    public static void main(String[] args) {
        int[] arr = {64, 34, 25, 12, 22, 11, 90};
        System.out.println("Original array: " +
Arrays.toString(arr));

        bubbleSort(arr);

        System.out.println("Sorted array: " +
Arrays.toString(arr));
    }
}
```

Output:

Original array: [64, 34, 25, 12, 22, 11, 90]
Sorted array: [90, 64, 34, 25, 22, 12, 11]

2. Selection Sort :

Selection Sort is a simple sorting algorithm that works by repeatedly finding the minimum element from the unsorted part of the array and swapping it with the first unsorted element. It effectively divides the array into two parts: the sorted part on the left and the unsorted part on the right.



Time Complexity:
Best case: O(n^2)
Worst case: O(n^2)

Where n is the size of the array.

Space Complexity:
O(1)

It is not a stable algorithm.

- **What is the output of the following Java program fragment:**

```
import java.util.Arrays;

public class Test {
  public static void selectionSort(int[] arr) {
    int n = arr.length;

    for (int i = 0; i < n - 1; i++) {
      int minIndex = i;

      for (int j = i + 1; j < n; j++) {
        if (arr[j] < arr[minIndex]) {
          minIndex = j;
        }
      }

      // Swap the minimum element with the first
unsorted element
      int temp = arr[i];
      arr[i] = arr[minIndex];
      arr[minIndex] = temp;
    }
  }

  public static void main(String[] args) {
    int[] arr = { 64, 34, 25, 12, 22, 11, 90 };
    System.out.println("Original array: " +
Arrays.toString(arr));
```

```
    selectionSort(arr);

    System.out.println("Sorted array: " +
Arrays.toString(arr));
  }
}
```

Output:

```
Original array: [64, 34, 25, 12, 22, 11, 90]
Sorted array: [11, 12, 22, 25, 34, 64, 90]
```

- **What is the output of the following Java program fragment:**

```
import java.util.Arrays;

public class Test {
    public static void selectionSortDescending(int[] arr) {
        int n = arr.length;

        for (int i = 0; i < n - 1; i++) {
            int maxIndex = i;

            for (int j = i + 1; j < n; j++) {
                if (arr[j] > arr[maxIndex]) {
                    maxIndex = j;
                }
            }

            // Swap the maximum element with the first
unsorted element
            int temp = arr[i];
            arr[i] = arr[maxIndex];
            arr[maxIndex] = temp;
        }
    }

    public static void main(String[] args) {
        int[] arr = {64, 34, 25, 12, 22, 11, 90};
        System.out.println("Original array: " +
Arrays.toString(arr));
```

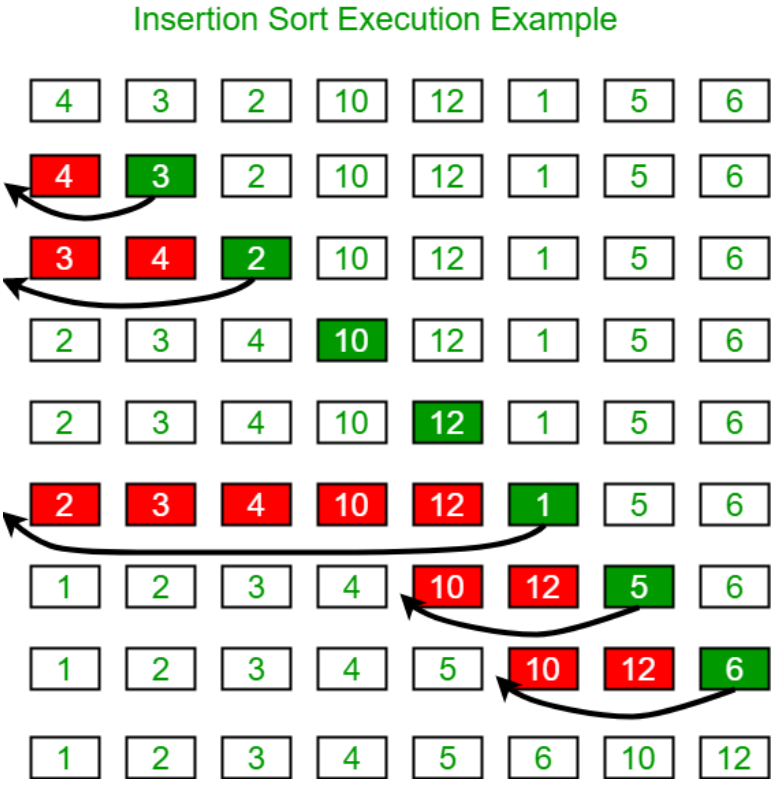
```
        selectionSortDescending(arr);

        System.out.println("Sorted array in descending
order: " + Arrays.toString(arr));
    }
}
```

Output:

```
Original array: [64, 34, 25, 12, 22, 11, 90]
Sorted array in descending order: [90, 64, 34, 25,
22, 12, 11]
```

3. Insertion Sort :

The Insertion Sort algorithm works by iterating through the array and "inserting" each element into its correct position in the already sorted part of the array.



Time Complexity:
Best case: O(n)
Worst case: O(n^2)

Where n is the size of the array.

Space Complexity:
O(1)

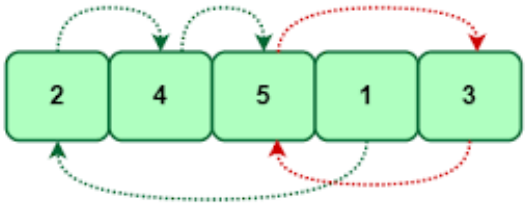
- **What is the output of the following Java program fragment:**

```
import java.util.Arrays;

public class Test{
   static void insertion(int[] arr){
      for(int i = 0; i < arr.length -1; i++){
         for(int j = i+1; j > 0; j--){
            if(arr[j-1] > arr [j]){
               int temp = arr[j];
               arr[j] = arr[j-1];
               arr[j-1] = temp;
            } else{
               break;
            }
         }
      }
   }

   public static void main(String[] args) {
      int[] arr = {0, -23, 56,18};
      insertion(arr);
      System.out.println(Arrays.toString(arr));
   }
}
```

Output:
[-23, 0, 18, 56]

4. Cycle Sort :

Cycle Sort is an in-place, unstable sorting algorithm that minimizes the number of writes performed during sorting. It is particularly useful for situations where writes to the data are more expensive than reads. The basic idea of Cycle Sort is to rotate the elements to their correct positions one by one, minimizing the number of swaps and writes.



Time Complexity:
Best case: O(n)
Worst case: O(n)

Where n is the size of the array.

Space Complexity:
O(1)

- **What is the output of the following Java program fragment:**

```
import java.util.Arrays;

public class Test {
  static void cycle(int[] arr) {
    int i = 0;
    while (i < arr.length) {
      if (arr[i] != i + 1) {
        int temp = arr[arr[i] - 1];
        arr[arr[i] - 1] = arr[i];
        arr[i] = temp;
      } else {
        i++;
      }
    }
  }

  public static void main(String[] args) {
    int[] arr = { 5, 2, 4, 3, 1 };
    cycle(arr);
    System.out.println(Arrays.toString(arr));
  }
}
```

Output:
[1, 2, 3, 4, 5]

- **What is the output of the following Java program fragment:**

```
import java.util.Arrays;

public class Hridi{
  static void cycle (int[] arr){
    for(int i = 0; i < arr.length ; ){
      if(arr[i] != i+1){
      int temp = arr[arr[i]-1];
      arr[arr[i]-1] = arr[i];
      arr[i] = temp;
      }
      else{
      i++;
      }
    }
  }

  public static void main(String[] args) {
    int[] arr = {5, 2, 4, 3, 1};
    cycle(arr);
    System.out.println(Arrays.toString(arr));
  }
}
```

Output:

[1, 2, 3, 4, 5]

- **What is the output of the following Java program fragment:**

```
import java.util.Arrays;

public class Test{
  static void cycle (int[] arr){
    for(int i = 0; i < arr.length ; ){
      if((arr[i] - 1) != i){
      int temp = arr[arr[i]-1];
      arr[arr[i]-1] = arr[i];
      arr[i] = temp;
      } else{
      i++;
      }
    }
  }

  public static void main(String[] args) {
    int[] arr = {5, 2, 4, 3, 1};
    cycle(arr);
    System.out.println(Arrays.toString(arr));
  }
}
```

Output:
[1, 2, 3, 4, 5]

- **What is the output of the following Java program fragment:**

```
import java.util.Arrays;

public class Hridi{
   static int cycle (int[] arr){
      int i = 0;
      while(i < arr.length ){
        if( arr[i] < arr.length && arr[i] != arr[arr[i]]){
         int temp = arr[arr[i]];
         arr[arr[i]] = arr[i];
         arr[i] = temp;
        } else {
         i++;
        }
      }

      for(int index = 0; index < arr.length; index++ ){
         if(arr[index] != index){
            return index;
         }
      }

      return arr.length;
   }

   public static void main(String[] args) {
      int[] arr = {5, 2, 4, 0, 1};
```

```
        int mNum = cycle(arr);
        System.out.println("Missing number is "+mNum);
    }
}
```

Output:
Missing number is 3

- **What is the output of the following Java program fragment:**

```
import java.util.Arrays;

public class Test{
    static void cycle (int[] arr){
        for(int i = 0; i < arr.length ; ){
            if(arr[i] < arr.length && arr[i] != i){
             int temp = arr[arr[i]];
             arr[arr[i]] = arr[i];
             arr[i] = temp;
            } else{
             i++;
            }
        }
    }

    public static void main(String[] args) {
        int[] arr = {5, 2, 0, 3, 1};
```

```
        cycle(arr);
         System.out.println(Arrays.toString(arr));
    }
}
```

Output:

[0, 1, 2, 3, 5]

6. String

String is a sequence of character.

Three types of String:
1. String,
2. StringBuffer,
3. StringBuilder.

- **What is the output of the following Java program fragment:**

```
public class Hamim {
    public static void main(String[] args) {
        String a = "Hamim Talukder";
        System.out.println("My name is "+a);
    }
}
```

Output:
My name is Hamim Talukder

- **What is the output of the following Java program fragment:**

```
import java.util.Arrays;

public class Test {
  public static void main(String[] args) {
    String a = "Hamim", b = "Hamim";
    System.out.println(a == b);
    String c = a;
    System.out.println(c==a);

    String name1 = new String("Hamim");
    String name2 = new String("Hamim");
    System.out.println(name1==name2);
    System.out.println(name1.equals(name2));
  }
}
```

Output:
true
true
false
true

1. String class:

Some methods related to String class:

- **length():** Returns the length of the string.
- charAt(int index): Returns the character at the specified index.
- **isEmpty():** Checks if the string is empty.
- **toUpperCase():** Converts the string to uppercase.
- **toLowerCase():** Converts the string to lowercase.
- **equals(String anotherString):** Compares the content of two strings for equality.
- **equalsIgnoreCase(String anotherString):** Compares the content of two strings for equality, ignoring case differences.
- **startsWith(String prefix):** Checks if the string starts with the specified prefix.
- **endsWith(String suffix):** Checks if the string ends with the specified suffix.
- **indexOf(String str):** Returns the index of the first occurrence of the specified substring.
- **lastIndexOf(String str):** Returns the index of the last occurrence of the specified substring.
- **substring(int beginIndex):** Returns a new string that is a substring of the original string, starting from the specified index.

- **substring(int beginIndex, int endIndex):** Returns a new string that is a substring of the original string, starting from the begin index and ending at the end index (exclusive).
- **replace(char oldChar, char newChar):** Replaces all occurrences of the specified old character with the new character.
- **replace(CharSequence target, CharSequence replacement):** Replaces all occurrences of the target string with the replacement string.
- **split(String regex):** Splits the string using the specified regular expression and returns an array of substrings.
- **trim():** Removes leading and trailing whitespaces from the string.
- **concat(String str):** Concatenates the specified string to the end of the original string.
- **compareTo(String anotherString):** Compares two strings lexicographically.
- **valueOf(dataType value):** Converts a primitive data type or object to a string representation.

- **length() :**

- **What is the output of the following Java program fragment:**

```
public class Test {
   public static void main(String[] args) {
      String str = "Hello, World!";
      int length = str.length();
      System.out.println("Length of the string: " + length);
   }
}
```

Output:
Length of the string: 13

- **charAt(int index) :**

- **What is the output of the following Java program fragment:**

```
public class Test {
   public static void main(String[] args) {
      String str = "Java";
      char ch = str.charAt(2);
      System.out.println("Character at index 2: " + ch);

   }
}
```

Output:
Character at index 2: v

- **isEmpty() :**

- **What is the output of the following Java program fragment:**

```
public class Test {
   public static void main(String[] args) {
      String str = "";
      boolean empty = str.isEmpty();
      System.out.println("Is the string empty? " + empty);
   }
}
```

Output:
Is the string empty? true

- **toUpperCase() :**

- **What is the output of the following Java program fragment:**

```
public class Test {
   public static void main(String[] args) {
      String str = "hello";
      String upperCaseStr = str.toUpperCase();
      System.out.println("Uppercase string: " +
upperCaseStr);
   }
}
```

Output:
Uppercase string: HELLO

- **toLowerCase() :**

- **What is the output of the following Java program fragment:**

```
public class Test {
   public static void main(String[] args) {
      String str = "WORLD";
      String lowerCaseStr = str.toLowerCase();
      System.out.println("Lowercase string: " +
lowerCaseStr);
   }
}
```

Output:
Lowercase string: world

- **equals(String anotherString) :**

- **What is the output of the following Java program fragment:**

```
public class Test {
   public static void main(String[] args) {
      String str1 = "Hello";
      String str2 = "Hello";
      boolean isEqual = str1.equals(str2);
      System.out.println("Are the strings equal? " +
isEqual);
   }
}
```

Output:
Are the strings equal? true

- **equalsIgnoreCase(String anotherString) :**

- **What is the output of the following Java program fragment:**

```
public class Test {
   public static void main(String[] args) {
      String str1 = "Hello";
      String str2 = "hello";
      boolean isEqualIgnoreCase =
str1.equalsIgnoreCase(str2);
      System.out.println("Are the strings equal
(ignore case)? " + isEqualIgnoreCase);
   }
}
```

Output:
Are the strings equal (ignore case)? true

- **startsWith(String prefix) :**

- **What is the output of the following Java program fragment:**

```
public class Test {
   public static void main(String[] args) {
      String str = "Hello, World!";
      boolean startsWith = str.startsWith("Hello");
      System.out.println("Does the string start with 'Hello'?
" + startsWith);
   }
}
```

Output:
Does the string start with 'Hello'? true

- **endsWith(String suffix) :**

- **What is the output of the following Java program fragment:**

```
public class Test {
   public static void main(String[] args) {
      String str = "Hello, World!";
      boolean endsWith = str.endsWith("World!");
      System.out.println("Does the string end with
'World!'? " + endsWith);
   }
}
```

Output:
Does the string end with 'World!'? true

- **indexOf(String str) :**

- **What is the output of the following Java program fragment:**

```
public class Test {
   public static void main(String[] args) {
      String str = "Java Programming";
      int index = str.indexOf("Programming");
      System.out.println("Index of 'Programming': "
+ index);
   }
}
```

Output:
Index of 'Programming': 5

- **lastIndexOf(String str) :**

- **What is the output of the following Java program fragment:**

```
public class Test {
   public static void main(String[] args) {
      String str = "Java Programming Java";
      int lastIndex = str.lastIndexOf("Java");
      System.out.println("Last index of 'Java': " + lastIndex);
   }
}
```

Output:
Last index of 'Java': 17

- **substring(int beginIndex) :**

- **What is the output of the following Java program fragment:**

```
public class Test {
   public static void main(String[] args) {
      String str = "Hello, World!";
      String subStr = str.substring(7);
      System.out.println("Substring from index 7: " +
subStr);
   }
}
```

Output:
Substring from index 7: World!

- **substring(int beginIndex, int endIndex) :**

- **What is the output of the following Java program fragment:**

```
public class Test {
  public static void main(String[] args) {
    String str = "Hello, World!";
    String subStr = str.substring(7, 12);
    System.out.println("Substring from index 7 to 12: " +
subStr);
  }
}
```

Output:
Substring from index 7 to 12: World

- **replace(char oldChar, char newChar) :**

- **What is the output of the following Java program fragment:**

```
public class Test {
    public static void main(String[] args) {
        String str = "Hello, World!";
        String newStr = str.replace('o', 'x');
        System.out.println("Replaced string: " + newStr);
    }
}
```

Output:
Replaced string: Hellx, Wxrld!

- **replace(CharSequence target, CharSequence replacement) :**

- **What is the output of the following Java program fragment:**

```
public class Test {
   public static void main(String[] args) {
      String str = "Hello, World!";
      String newStr = str.replace("World", "Universe");
      System.out.println("Replaced string: " + newStr);
   }
}
```

Output:
Replaced string: Hello, Universe!

- **split(String regex) :**

- **What is the output of the following Java program fragment:**

```
public class Test {
   public static void main(String[] args) {
      String str = "apple,orange,banana";
      String[] fruits = str.split(",");
      for (String fruit : fruits) {
         System.out.println(fruit);
      }
   }
}
```

Output:
apple
orange
banana

- **trim() :**

- **What is the output of the following Java program fragment:**

```
public class Test {
   public static void main(String[] args) {
      String str = "   Hello, World!   ";
      String trimmedStr = str.trim();
      System.out.println("Trimmed string: '" +
trimmedStr + "'");
   }
}
```

Output:
Trimmed string: 'Hello, World!'

- **concat(String str) :**

- **What is the output of the following Java program fragment:**

```
public class Test {
   public static void main(String[] args) {
      String str1 = "Hello";
      String str2 = "World";
      String result = str1.concat(" " + str2);
      System.out.println("Concatenated string: " + result);
   }
}
```

Output:
Concatenated string: Hello World

- **compareTo(String anotherString) :**

- **What is the output of the following Java program fragment:**

```
public class Test {
   public static void main(String[] args) {
      String str1 = "apple";
      String str2 = "banana";
      int comparisonResult = str1.compareTo(str2);
      System.out.println("Comparison result: " +
comparisonResult);
   }
}
```

Output:
Comparison result: -1

- **contains(String anotherString) :**

- **What is the output of the following Java program fragment:**

```
public class Test {
  public static void main(String[] args) {
    String s1 = "Hamim Talukder 1.O";
    String s2 = new String("Hamim Talukder 1.O");
    System.out.println("s1 = " + s1);
    System.out.println("s2 = " + s2);
    if (s1.contains(s2)) {
      System.out.println("Equals");
    } else {
```

```
    System.out.println("Not equals");
   }
  }
}
```

Output:
s1 = Hamim Talukder 1.O
s2 = Hamim Talukder 1.O
Equals

- **What is the output of the following Java program fragment:**

```
public class Test {
   public static void main(String[] args) {
      String s1 = "Hamim Talukder 1.O";
      System.out.println("s1 = " + s1);
      if (s1.contains("Hamim")) {
         System.out.println("Equals");
      } else {
         System.out.println("Not equals");
      }
   }
}
```

Output:
s1 = Hamim Talukder 1.O
Equals

- **valueOf(dataType value) :**

- **What is the output of the following Java program fragment:**

```
public class Test {
   public static void main(String[] args) {
      int num = 42;
      String str = String.valueOf(num);
      System.out.println("String representation of
the number: " + str);
   }
}
```

Output:
String representation of the number: 42

- **What is the output of the following Java program fragment:**

```
package basicjava;
import java.util.Scanner;

public class JavaCode {
   public static void main(String[] args) {
      Scanner scan = new Scanner(System.in);
      String s1 = "Hamim Talukder 1.O";
      String s2 = new String("Hamim Talukder 2.O");
      char s3[ ] = {'H','a','m','i','m'};
      System.out.println("s1 = "+s1);
      System.out.println("s2 = "+s2);
      System.out.println(s3);
   }
}
```

Output:

```
s1 = Hamim Talukder 1.O
s2 = Hamim Talukder 2.O
Hamim
```

- **What is the output of the following Java program fragment:**

```
package basicjava;
import java.util.Scanner;

public class JavaCode {
   public static void main(String[] args) {
      Scanner scan = new Scanner(System.in);
      String s1 = "Hamim Talukder 1.O";
      String s2 = new String("Hamim Talukder 2.O");
      System.out.println("s1 = "+s1);
      System.out.println("s2 = "+s2);
      int len = s1.length();
      System.out.println("Length of s1 = "+len);
   }
}
```

Output:

```
s1 = Hamim Talukder 1.O
s2 = Hamim Talukder 2.O
Length of s1 = 18
```

- **Make a program that will show whether two Strings are equals or not.**

```
package basicjava;
import java.util.Scanner;

public class JavaCode {
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        String s1 = "Hamim Talukder 1.O";
        String s2 = new String("Hamim Talukder 1.O");
        System.out.println("s1 = "+s1);
        System.out.println("s2 = "+s2);
        if(s1.equals(s2)){
            System.out.println("Equals");
        }
        else{
            System.out.println("Not equals");
        }
    }
}
```

or,

```
package basicjava;
import java.util.Scanner;

public class JavaCode {
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        String s1 = "Hamim Talukder 1.O";
        String s2 = new String("Hamim Talukder 1.O");
```

```
      System.out.println("s1 = "+s1);
       System.out.println("s2 = "+s2);
       if(s1.contains(s2)){
          System.out.println("Equals");
       }
       else{
          System.out.println("Not equals");
       }
   }
}
```

Output:
s1 = Hamim Talukder 1.O
s2 = Hamim Talukder 1.O
Equals

- **What is the output of the following Java program fragment:**

```
package basicjava;
import java.util.Scanner;

public class JavaCode {
   public static void main(String[] args) {
      Scanner scan = new Scanner(System.in);
      String s1 = "Hamim Talukder 1.O";
      String s2 = new String("Hamim Talukder 1.O");
      System.out.println("s1 = "+s1);
      System.out.println("s2 = "+s2);
      if(s1.contains("Hami")){
         System.out.println("Equals");
      }
```

```
        else{
            System.out.println("Not equals");
        }
    }
}
```

Output:
s1 = Hamim Talukder 1.O
s2 = Hamim Talukder 1.O
Equals

Here,
contains( ) method check which parameter we have given to the contains method is available in s1 string or not.

- **What is the output of the following Java program fragment:**

```
package basicjava;
import java.util.Scanner;

public class JavaCode {
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        String s1 = "hamim Talukder 1.O";
        String s2 = new String("Hamim Talukder 1.O");
        System.out.println("s1 = "+s1);
        System.out.println("s2 = "+s2);
        if(s1.equalsIgnoreCase(s2)){
            System.out.println("Equals");
        }
```

```
        else{
            System.out.println("Not equals");
        }
    }
}
```

Output:

```
s1 = hamim Talukder 1.O
s2 = Hamim Talukder 1.O
Equals
```

Here,
equalsIgnoreCase( ) method can compare two Strings ignoring their case(capital letter/small ietter).

- **What is the output of the following Java program fragment:**

```
package basicjava;
import java.util.Scanner;

public class JavaCode {
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        String s1 = "hamim Talukder 1.O";
        String s2 = new String("Hamim Talukder 1.O");
        System.out.println("s1 = "+s1);
        System.out.println("s2 = "+s2);
        boolean con = s1.contains("Taluk");
        System.out.println(con);
    }
}
```

Output:
s1 = hamim Talukder 1.O
s2 = Hamim Talukder 1.O
true

- **What is the output of the following Java program fragment:**

```
package basicjava;
import java.util.Scanner;

public class JavaCode {
   public static void main(String[] args) {
      Scanner scan = new Scanner(System.in);
      String s1 = "";
      String s2 = new String("Hamim Talukder 1.O");
      System.out.println("s1 = "+s1);
      System.out.println("s2 = "+s2);
      boolean b = s1.isEmpty();
      System.out.println("S1 string is empty = "+b);
   }
}
```

Output:
s1 =
s2 = Hamim Talukder 1.O
S1 string is empty = true

- **Make a program that will show sum of two Strings.**

```
package basicjava;
import java.util.Scanner;

public class JavaCode {
   public static void main(String[] args) {
      Scanner scan = new Scanner(System.in);
      String firstName = "Hamim";
      String lastName = " Talukder";
      String fullName = firstName + lastName;
      System.out.println("Full Name = "+fullName);
   }
}
```

Output:
Full Name = Hamim Talukder

- **What is the output of the following Java program fragment:**

```
package basicjava;
import java.util.Scanner;

public class JavaCode {
   public static void main(String[] args) {
      Scanner scan = new Scanner(System.in);
      String firstName = "Hamim";
      String lastName = " Talukder";
      String fullName = firstName + lastName;
      System.out.println("Full Name = "+fullName + 25);
   }
}
```

Output:
Full Name = Hamim Talukder25

- **Make a program that will concat two Strings.**

```
package basicjava;
import java.util.Scanner;

public class JavaCode {
   public static void main(String[] args) {
      Scanner scan = new Scanner(System.in);
      String firstName = "Hamim";
      String lastName = " Talukder";
      String fullName = firstName.concat(lastName);
      System.out.println("Full Name = "+fullName);
   }
}
```

Output:
Full Name = Hamim Talukder

Here,
We have used concat( ) method to join two Strings.

- **Make a program that will convert any String into uppercase String.**

```
package basicjava;
import java.util.Scanner;

public class JavaCode {
   public static void main(String[] args) {
      Scanner scan = new Scanner(System.in);
      String fullName = "Hamim Talukder";
      String upperName = fullName.toUpperCase();
      System.out.println("Full Name = "+upperName);
   }
}
```

Output:

Full Name = HAMIM TALUKDER

- **Make a program that will convert any String into lowercase String.**

```
package basicjava;
import java.util.Scanner;

public class JavaCode {
   public static void main(String[] args) {
      Scanner scan = new Scanner(System.in);
      String fullName = "Hamim Talukder";
      String lowerName = fullName.toLowerCase();
      System.out.println("Full Name = "+lowerName);
   }
}
```

Output:
Full Name = hamim talukder

- **What is the output of the following Java program fragment:**

```
package basicjava;
import java.util.Scanner;

public class JavaCode {
   public static void main(String[] args) {
      Scanner scan = new Scanner(System.in);
      String fullName = "Hamim Talukder";
      boolean start = fullName.startsWith("Hami");
      boolean end = fullName.endsWith("r");
      System.out.println("This String is starts with
Hami : "+start);
      System.out.println("This String is ends with r :
"+end);
   }
}
```

Output:
This String is starts with Hami : true
This String is ends with r : true

Here,
We have used startsWith( ) and endsWith( ) method.

- **What is the output of the following Java program fragment:**

```
package basicjava;
import java.util.Scanner;

public class JavaCode {
   public static void main(String[] args) {
      Scanner scan = new Scanner(System.in);
      String names[ ] = {"Hamim
Talukder","Jim","Rahim","Mim","Mithu"};
      for (int i = 0; i < 5; i++) {
         System.out.println(i+1+" no. name = "+names[i]);
      }
   }
}
```

Output:

```
1 no. name = Hamim Talukder
2 no. name = Jim
3 no. name = Rahim
4 no. name = Mim
5 no. name = Mithu
```

- **What is the output of the following Java program fragment:**

```
package basicjava;
import java.util.Scanner;

public class JavaCode {
   public static void main(String[] args) {
      Scanner scan = new Scanner(System.in);
      String country = "Bangladesh is my country";
      System.out.println(country);
      char ch = country.charAt(0);
      System.out.println("ch = "+ch);
      int value = country.codePointAt(0);
      System.out.println("ASCII value of o index : "+value);
      int pos = country.indexOf('n');
      System.out.println("First position of n : "+pos);
      pos = country.lastIndexOf('n');
      System.out.println("Last position of n : "+pos);
      pos = country.indexOf("is");
      System.out.println("First position of is : "+pos);
   }
}
```

Output:

```
Bangladesh is my country
ch = B
ASCII value of o index : 66
First position of n : 2
Last position of n : 20
First position of is : 11
```

Here,
We have used :
charAt( ) method for watching the character of any index,
codePoint( ) method for watching the ASCII value of any character.
indexOf( ) method for watching the first index of any character of the String,
latsIndexOf( ) method for watching the last character of the String.

- **What is the output of the following Java program fragment:**

```
package basicjava;
import java.util.Scanner;

public class JavaCode {
   public static void main(String[] args) {
      Scanner scan = new Scanner(System.in);
      String country = "  Bangladesh  is  my  country  ";
      System.out.println(country);
      String s3 = country.trim();
      System.out.println(s3);
   }
}
```

Output:

```
  Bangladesh  is  my  country
Bangladesh  is  my  country
```

Here,
trim( ) method removes all the space before the first character and after the last character of a String.

- **What is the output of the following Java program fragment:**

```
package basicjava;
import java.util.Scanner;

public class JavaCode {
   public static void main(String[] args) {
      Scanner scan = new Scanner(System.in);
      String s1 = "This is my country";
      System.out.println("Before replacing : "+s1);
      String s2 = s1.replace('i', 'j');
      System.out.println("After replacing : "+s2);
      String[] s3 = s1.split(" ");
      System.out.println("After split by space : ");
      for (String x : s3) {
         System.out.println(x);
      }
   }
}
```

Output:

```
Before replacing : This is my country
After replacing : Thjs js my country
After split by space :
This
is
my
country
```

Here,
In the String 'i' character is replaced by 'j' character using replace( ) method. split( ) method is used to split any String.

- **What is the output of the following Java program fragment:**

```
package basicjava;
import java.util.Scanner;

public class JavaCode {
   public static void main(String[] args) {
      Scanner scan = new Scanner(System.in);
      String s1 = "Hamim Talukder";
      System.out.print("Before replace : ");
      System.out.println(s1);
      System.out.print("After replace : ");
      s1.replace('H', 'm');
      System.out.println(s1);
   }
}
```

Output:
Before replace : Hamim Talukder
After replace : Hamim Talukder

Here,
We can see that String can not be changed in the same variable.

- **What is the output of the following Java program fragment:**

```
package basicjava;
import java.util.Scanner;

public class JavaCode {
   public static void main(String[] args) {
      Scanner scan = new Scanner(System.in);
      String s1 = "Hamim Talukder";
      System.out.print("Before replace : ");
      System.out.println(s1);
      s1 = s1.replace('H', 'm');
      System.out.print("After replace : ");
      System.out.println(s1);
   }
}
```

Output:

Before replace : Hamim Talukder
After replace : mamim Talukder

2. StringBuffer class:

StringBuffer is a class in Java that represents a mutable sequence of characters. It provides an alternative to the immutable String class, allowing you to modify the contents of a string without creating a new object every time.

Some methods related to StringBuffer class:

- setCharAt()
- append( )
- reverse( )
- delete( )
- setLength( )

- **setCharAt(index, char ch)**

```
public class ReplaceCharacterInStringBufferExample {
  public static void main(String[] args) {
    StringBuffer strBuffer = new StringBuffer("Hello,
World!");

    strBuffer.setCharAt(7, 'X');

    System.out.println("Modified StringBuffer: " +
strBuffer.toString());
  }
}
```

Output:
Modified String: Hello, WXrld!

- **append(any types of data) :**

```
package basicjava;
import java.util.Scanner;

public class JavaCode {
   public static void main(String[] args) {
      Scanner scan = new Scanner(System.in);
      StringBuffer s1 = new StringBuffer("Hamim");
      System.out.println("Before add other string : ");
      System.out.println(s1);
      s1.append("Talukder");
      s1.append(25);
      System.out.println("After add other content : ");
      System.out.println(s1);
   }
}
```

Output:
Before add other string :
Hamim
After add other content :
HamimTalukder25

Here,
We have used StringBuffer data type so that we can make changes in the same String variable.
We also used append( ) method to add Strings.

- **reverse() :**

```
package basicjava;
import java.util.Scanner;

public class JavaCode {
   public static void main(String[] args) {
      Scanner scan = new Scanner(System.in);
      StringBuffer s1 = new StringBuffer("Hamim");
      System.out.println("Before reverse : ");
      System.out.println(s1);
      s1.reverse();
      System.out.println("After reverse : ");
      System.out.println(s1);
   }
}
```

Output:

```
Before reverse :
Hamim
After reverse :
mimaH
```

Here,
We have used reverse( ) method for reversing String.

- **delete(char startChar, char EndChar) :**

- **Make a program that will delete String's character from m to n.**

```
package basicjava;
import java.util.Scanner;

public class JavaCode {
   public static void main(String[] args) {
      Scanner scan = new Scanner(System.in);
      StringBuffer s1 = new StringBuffer("Hamim");
      System.out.println("Before deleting characters : ");
      System.out.println(s1);
      s1.delete(0,4);
      System.out.println("After deleting characters : ");
      System.out.println(s1);
   }
}
```

Output:

```
Before deleting characters :
Hamim
After deleting characters :
m
```

Here,
We have used delete( ) method to delete characters.

- **setLength(int length) :**

```
package basicjava;
import java.util.Scanner;

public class JavaCode {
   public static void main(String[] args) {
      Scanner scan = new Scanner(System.in);
      StringBuffer s1 = new StringBuffer("Hamim
Talukder");
      System.out.println(s1);
      s1.setLength(5);
      System.out.println(s1);
   }
}
```

Output:
Hamim Talukder
Hamim

Here
We have used setLength( ) method for trimming String.

- **What is the output of the following Java program fragment:**

```
package basicjava;
import java.util.Scanner;

public class JavaCode {
   public static void main(String[] args) {
      Scanner scan = new Scanner(System.in);
      String s = "Hamim Talukder";
      StringBuffer s1 = new StringBuffer(s);
      System.out.println(s1);
   }
}
```

Output:
Hamim Talukder

- **Make a program that will show String palindrome.**

```
package basicjava;
import java.util.Scanner;

public class JavaCode {
   public static void main(String[] args) {
      Scanner scan = new Scanner(System.in);
      String s1 = "madam";
      StringBuffer s2 = new StringBuffer(s1);
      String s3 = s2.reverse().toString();
      if(s1.equals(s3)){
         System.out.println("Palindrome");
      }
```

```
        else{
            System.out.println("Not a palindrome");
        }
    }
}
```

Output:
Palindrome

Here,
We have used toString( ) method to assign StringBuffer value in String variavle.

3. StringBuilder class:

In Java, StringBuilder is a class that provides a convenient way to work with strings when you need to perform multiple concatenations or modifications. Unlike the standard String class, which is immutable (meaning its value cannot be changed after it's created), StringBuilder is mutable, allowing you to modify the contents of the string without creating new objects.

Some methods related to StringBuilder class:

- setCharAt()
- append( )
- reverse( )
- delete( )
- setLength( )

- **setCharAt(index, char ch)**

```
public class ReplaceCharacterAtIndexExample {
  public static void main(String[] args) {
    StringBuilder str = new StringBuilder("Hello, World!");

    str.setCharAt(7, 'X');

    System.out.println("Modified String: " + str.toString());
  }
}
```

Output:
Modified String: Hello, WXrld!

- **append(any types of data) :**

```
package basicjava;
import java.util.Scanner;

public class JavaCode {
   public static void main(String[] args) {
      Scanner scan = new Scanner(System.in);
      StringBuilder str = new StringBuilder("Hamim");
      System.out.println("str = "+str);
      str.append(" Talukder");
      str.append(23);
      str.append( 12.5);
      System.out.println("str = "+str);
   }
}
```

Output:
str = Hamim
str = Hamim Talukder2312.5

Here,
We have used StringBuilder data type.
StringBuilder datatype is quite similer to
StringBuffer data type.

- **reverse() :**

```
package basicjava;
import java.util.Scanner;

public class JavaCode {
   public static void main(String[] args) {
      Scanner scan = new Scanner(System.in);
      StringBuilder str = new
StringBuilder("Hamim");
      str.append(" Talukder");
      System.out.println("str = "+str);
      str.reverse();
      System.out.println("str = "+str);
   }
}
```

Output:
str = Hamim Talukder
str = redkulaT mimaH

Here,
We have used reverse( ) method for reversing String.

- **delete(char startChar, char EndChar) :**

- **What is the output of the following Java program fragment:**

```
package basicjava;
import java.util.Scanner;

public class JavaCode {
   public static void main(String[] args) {
      Scanner scan = new Scanner(System.in);
      StringBuilder str = new StringBuilder("Hamim");
      str.append(" Talukder");
      System.out.println("str = "+str);
      str.delete(2, 4);
      System.out.println("str = "+str);
   }
}
```

Output:

```
str = Hamim Talukder
str = Ham Talukder
```

Here,
We have used delete( ) method to delete characters.

- **setLength(int length) :**

```
import java.util.Scanner;

public class Test {
  public static void main(String[] args) {
    Scanner scan = new Scanner(System.in);
    StringBuilder s1 = new StringBuilder("Hamim
Talukder");
    System.out.println(s1);
    s1.setLength(5);
    System.out.println(s1);
  }
}
```

Output:
Hamim Talukder
Hamim

Here
We have used setLength( ) method for trimming String.

- **toString Method:**

The toString() of StringJoiner is used to convert StringJoiner to String. It returns the current value, consisting of the prefix, the values added so far separated by the delimiter, and the suffix, unless no elements have been added in which case, the prefix + suffix or the emptyValue characters are returned

Syntax:

public String toString()

Returns: This method returns the string representation of this StringJoiner

- **What is the output of the following Java program fragment:**

```
package FirstPackage;

import java.util.StringJoiner;

public class Test {
   public static void main(String[] args)
   {

      StringJoiner str = new StringJoiner(" ");

      str.add("Geeks");
      str.add("for");
      str.add("Geeks");

      System.out.println("StringJoiner: "
            + str.toString());
   }
}
```

Output:
StringJoiner: Geeks for Geeks

- **What is the output of the following Java program fragment:**

```
package FirstPackage;

import java.util.StringJoiner;

public class Test {
   public static void main(String[] args)
   {

      StringJoiner str = new StringJoiner(", ");

      // Adding elements in the StringJoiner
      str.add("Geeks");
      str.add("for");
      str.add("Geeks");
      str.add("A");
      str.add("Computer");
      str.add("Portal");

      // Print the StringJoiner
      // using toString() method
      System.out.println("StringJoiner: "
            + str.toString());
   }
}
```

Output:
StringJoiner: Geeks, for, Geeks, A, Computer, Portal

- **What is the output of the following Java program fragment:**

```
package FirstPackage;

public class Person {
   String name;
   int age;
   Person (String name, int age){
      this.name = name;
      this.age = age;
   }

   @Override
   public String toString(){
      return  "Name : "+ name + "\nAge : " +age;
   }
}
```

```
package FirstPackage;

import java.util.Scanner;
import java.util.SortedMap;

public class Test {
   public static void main(String[] args){

   Person p1 = new Person("HaMeem Talukdar",23);
   Person p2 = new Person("Unknown Person",23);

      System.out.println(p1);
      System.out.println(p2);
   }
}
```

Output:
Name : HaMeem Talukdar
Age : 23
Name : Unknown Person
Age : 23

Here,
toString() is the method of object class. That's it has been overridden.

- **String comparison equal( ) and ==**

Main difference between == and equals() in Java is that "==" is used to compare primitive while equals() method is recommended to check equality of objects.

- **equals() :**

Compares the original content of the string.

- **What is the output of the following Java program fragment:**

```
package FirstPackage;

public class Test {
   public static void main(String[] args)
   {
      String password1 = "HaMeem2011";
      String password2 = "HaMeem2011";
      String password3 = new String("HaMeem2011");
      String password4 = new String("HaMeem2011");


System.out.println(password1.equals(password2));

System.out.println(password1.equals(password3));

System.out.println(password3.equals(password4));
   }
}
```

Output:
true
true
true

- **== operator:**

Compares the references of the objects not the value.

- **What is the output of the following Java program fragment:**

```
package FirstPackage;

public class Test {
   public static void main(String[] args)
   {
      String password1 = "HaMeem2011";
      String password2 = "HaMeem2011";
      String password3 = new String("HaMeem2011");
      String password4 = new String("HaMeem2011");

      System.out.println(password1==password2);
      System.out.println(password1==password3);
      System.out.println(password3==password4);
   }
}
```

Output:
true
false
false

- **wrapper class :**

Wrapper classes are used to convert primitive data type into object and object into primitive data type.

primitive	Wrapper class
boolean	Boolean
char	Character
byte	Byte
short	Short
int	Integer
long	Long
float	Float
double	Double

- Autoboxing = converting primitive to object
- Unboxing = converting object to primitive

- **What is the output of the following Java program fragment:**

```
package basicjava;
import java.util.Scanner;

public class JavaCode {
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        //primitive to object
        int x = 30;
        Integer y = Integer.valueOf(x);
        System.out.println("y = "+y);
    }
}
```

or,

```
package basicjava;
import java.util.Scanner;

public class JavaCode {
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        //primitive to object
        int x = 30;
        Integer y = x;//Integer.valueOf(x) ->
autoboxing
        System.out.println("y = "+y);
    }
}
```

Output:

```
y = 30
```

Here,
We have converted primitive data type into object (Autoboxing).

- **What is the output of the following Java program fragment:**

```
package basicjava;
import java.util.Scanner;

public class JavaCode {
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        //object to primitiveb data type
        Double d = new Double(10.25);
        System.out.println("d = "+d);
        double e = d.doubleValue();
        System.out.println("e = "+e);
    }
}
```

or,

```
package basicjava;
import java.util.Scanner;

public class JavaCode {
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        //object to primitiveb data type
        Double d = new Double(10.25);
        System.out.println("d = "+d);
        double e = d; //d.doubleValue()
        System.out.println("e = "+e);
    }
}
```

Output:
d = 10.25
e = 10.25

Here,
We have converted object into primitive data type (Unboxing).

Conversion between String and Primitive Data type

- Converting primitive to String data type :

- **Write a program that will convert a primitive data type into String data type..**

```
package basicjava;
import java.util.Scanner;

public class JavaCode {
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        //primitive to String data type
        int i = 100;
        String s = Integer.toString(i);
        System.out.println("s = "+s);
    }
}
```

Output:
s = 100

- **What is the output of the following Java program fragment:**

```
package basicjava;
import java.util.Scanner;

public class JavaCode {
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        //primitive to String data type
        double i = 100;
```

```
        String s = Double.toString(i);
        System.out.println("s = "+s);
    }
}
```

Output:
s = 100.0

- **What is the output of the following Java program fragment:**

```
package basicjava;
import java.util.Scanner;

public class JavaCode {
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        //primitive to String data type
        boolean i = true;
        String s = Boolean.toString(i);
        System.out.println("s = "+s);
    }
}
```

Output:
s = true

- **Converting String to Primitive data type :**

- **What is the output of the following Java program fragment:**

```
package basicjava;
import java.util.Scanner;

public class JavaCode {
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        //String to primitive data type
        String s = "32";
        int i = Integer.parseInt(s);
        System.out.println("i  = "+i);
    }
}
```

or,

```
package basicjava;
import java.util.Scanner;

public class JavaCode {
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        //String primitive data type
        String s = "32";
        int i = Integer.valueOf(s);
        System.out.println("i  = "+i);
    }
}
```

Output:
i = 32

- **What is the output of the following Java program fragment:**

```
package basicjava;
import java.util.Scanner;

public class JavaCode {
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        //String to primitive data type
        String s = "32";
        double i = Double.parseDouble(s);
        System.out.println("i  = "+i);
    }
}
```

Output:
i = 32.0

- **Write a program that will convert decimal To binary, octal, and hexadecimal.**

```
package basicjava;
import java.util.Scanner;

public class JavaCode {
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        int decimal = 15;
        String binary =
Integer.toBinaryString(decimal);
        System.out.println("Binary form of 15 =
"+binary);
        String octal = Integer.toOctalString(decimal);
        System.out.println("Octal form of 15 =
"+octal);
        String hexa = Integer.toHexString(decimal);
        System.out.println("Hexadecimal form of 15
= "+hexa);
    }
}
```

Output:
Binary form of 15 = 1111
Octal form of 15 = 17
Hexadecimal form of 15 = f

- **Write a program that will convert binary, octal, and hexadecimal to decimal.**

```
package basicjava;
import java.util.Scanner;

public class JavaCode {
   public static void main(String[] args) {
      Scanner scan = new Scanner(System.in);
      String binary = "1010";
      Integer decimal = Integer.parseInt(binary, 2);
      System.out.println("Decimal = "+decimal);
      String octal = "675";
      decimal = Integer.parseInt(octal, 8);
      System.out.println("Octal = "+decimal);
      String hexa = "A";
      decimal = Integer.parseInt(hexa, 16);
      System.out.println("Hexadecimal =
"+decimal);
   }
}
```

Output:
Decimal = 10
Octal = 445
Hexadecimal = 10

Here,
The first parameter of parseInt( ) method take the binary number as a String and second parameter is the base number <parseInt(binary, 2)>.

- **Write a program that will show date.**

```
package basicjava;
import java.util.Date;
import java.util.Scanner;

public class JavaCode {
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        Date date = new Date();
        System.out.println("Date is : "+date);
    }
}
```

Output:

```
ate is : Sun Jan 29 04:05:08 BDT 2023
```

or,

```
package basicjava;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Scanner;

public class JavaCode {
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        Date date = new Date();
        DateFormat dateFormat = new
SimpleDateFormat("dd/MM/YYYY");
        String currentDate = dateFormat.format(date);
        System.out.println("Current date : "+currentDate);
    }
}
```

Output:
Current date : 29/01/2023

Here,
Date is a class and we declare date as an object/variavle of the Date class.
We also declare dataFormat variable of the DateFormat class for formatting the date.

- **Write a program that will show time.**

```
package basicjava;
import java.time.LocalTime;
import java.util.Scanner;

public class JavaCode {
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        LocalTime time = LocalTime.now();
        System.out.println("Time is : "+time);
    }
}
```

Output:
Time is : 04:29:13.131902400

or,

```
package basicjava;
import java.time.LocalTime;
import java.time.format.DateTimeFormatter;
import java.util.Scanner;

public class JavaCode {
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        LocalTime time = LocalTime.now();
        DateTimeFormatter fomatter =
DateTimeFormatter.ofPattern("hh:mm:ss");
        String formattedTime = time.format(fomatter);
        System.out.println("Time is : "+formattedTime);
    }
}
```

Output:

```
Time is : 04:34:11
```

- **Write a program that will create random number 0 to 9.**

```
package basicjava;
import java.util.Random;
import java.util.Scanner;

public class JavaCode {
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        Random rand = new Random();
        int randomNumber = rand.nextInt(10); // 0 to 9
        System.out.println("Random number =
"+randomNumber);
    }
}
```

Output:
Random number = 5

Here,
We have used Random class to create random number.

or,

```
package basicjava;
import java.util.Scanner;

public class JavaCode {
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
```

```
int randomNumber = (int) (Math.random()*10); // 0 to 9
        System.out.println("Random number = "+randomNumber);
    }
}
```

Output:
Random number = 7

Here,
We have used random method of Math class to create random number.

- **Write a program that will create random number from 1 to 10.**

```
package basicjava;
import java.util.Random;
import java.util.Scanner;

public class JavaCode {
   public static void main(String[] args) {
      Scanner scan = new Scanner(System.in);
      Random rand = new Random();
      int randomNumber = rand.nextInt(10) + 1; // 1 to
10
      System.out.println("Random number =
"+randomNumber);
   }
}
```

Output:
Random number = 1

or,

```
package basicjava;
import java.util.Scanner;

public class JavaCode {
   public static void main(String[] args) {
      Scanner scan = new Scanner(System.in);
      int randomNumber = (int) (Math.random()*10) +
1; // 1 to 10
      System.out.println("Random number =
"+randomNumber);
   }
}
```

Output:

Random number = 3

5. Trie

A Trie (pronounced "try"), also known as a prefix tree or digital tree or k-ary search tree or retrieval tree, is a multiway tree data structure used for storing strings over an alphabet. It is used to store a large amount of strings. The pattern matching can be done efficiently using tries.

It's particularly useful for tasks involving prefix-based operations, such as searching for words with specific prefixes or autocomplete suggestions.

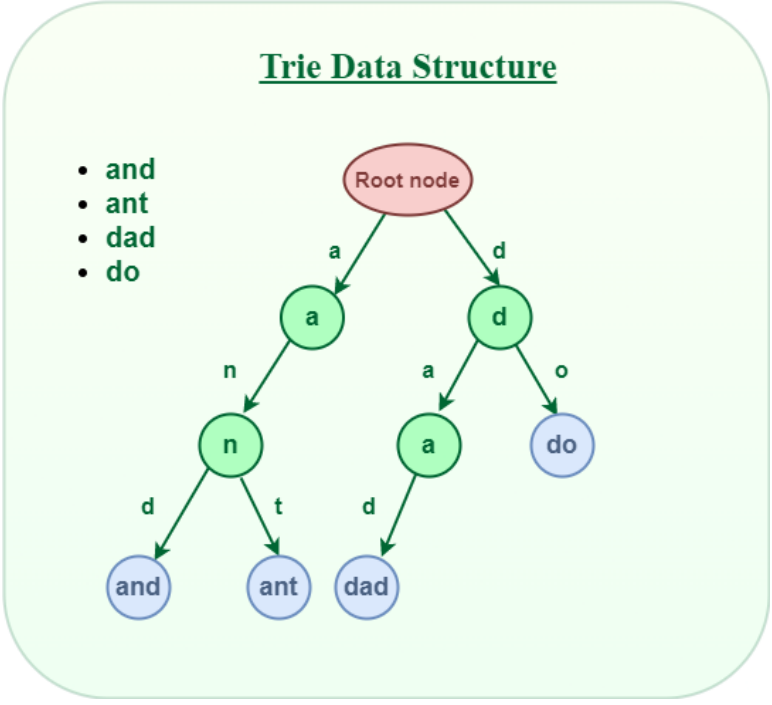
The trie shows words like allot, alone, ant, and, are, bat, bad. The idea is that all strings sharing common prefix should come from a common node. The tries are used in spell checking programs.

- Preprocessing pattern improves the performance of pattern matching algorithm. But if a text is very large then it is better to preprocess text instead of pattern for efficient search.
- A trie is a data structure that supports pattern matching queries in time proportional to the pattern size.

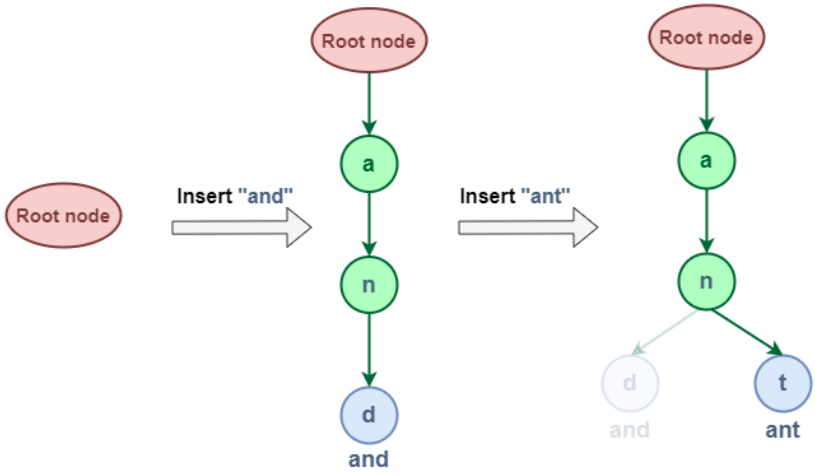
If we store keys in a binary search tree, a well balanced BST will need time proportional to M * log N, where M is the maximum string length and N is the number of keys in the tree. Using Trie, the

key can be searched in O(M) time. However, the penalty is on Trie storage requirements (Please refer to Applications of Trie for more details).

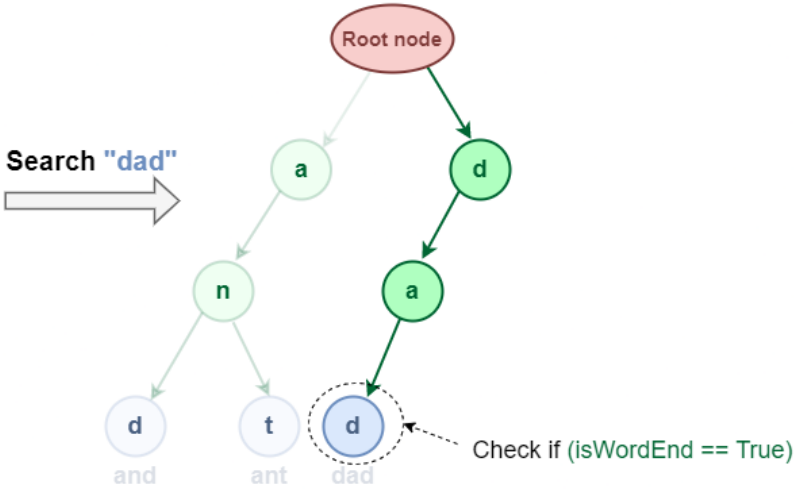
Trie is also known as digital tree or prefix tree. Refer to this article for more detailed information.



Insert Operation in Trie:



Search Operation in Trie:



Time complexity:
O(key_length)

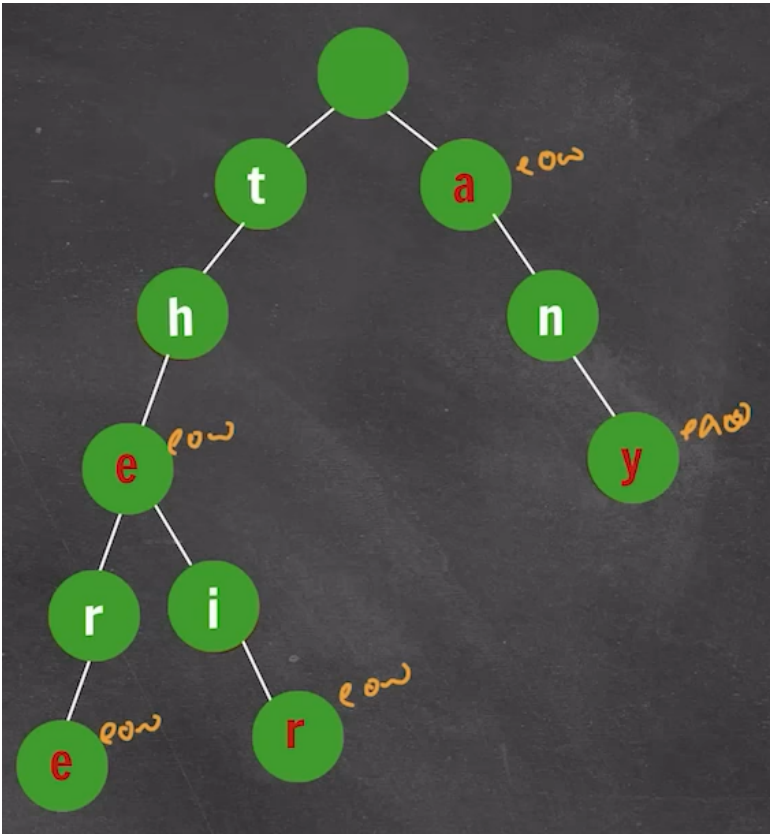
Space complexity:
O(ALPHABET_SIZE * key_length * N)

where **N** is the **number of keys in Trie**. There are efficient representations of trie nodes (e.g. compressed trie, ternary search tree, etc.) to minimize the memory requirements of the trie.

- **Trie code implementation:**

words[ ] = “the”, “a”, “there”, “their”, “any”

Trie of these words :



```
public class Test {
  static class Node {
     Node[] children;
     boolean eow; // endOfWord

     public Node() {
        children = new Node[26];
        for (int i=0; i<26; i++) {
           children[i] = null;
        }
     }
  }

  public static Node root = new Node();

  public static void insert(String word) { //O(wordLength)
     Node curr = root;

     for(int i = 0; i<word.length(); i++) {
        int idx = word.charAt(i)-'a';
        if(curr.children[idx] == null) {
           curr.children[idx] = new Node();
        }
        curr = curr.children[idx];
     }
     curr.eow = true;
  }

  public static boolean search(String key) {
//O(wordLength)
     Node curr = root;
```

```
        for(int i = 0; i<key.length(); i++) {
            int idx = key.charAt(i)-'a';
            if(curr.children[idx] == null) {
                return false;
            }
            curr = curr.children[idx];
        }

        return curr.eow == true;
    }

    public static void main(String args[]) {
        String words[] = {"the", "a", "there", "their", "any",
"thee"};

        for (String word : words) {
            insert(word);
            System.out.println("inserted " + word);
        }

        System.out.println("thee -> " + search("thee"));
        System.out.println("thor -> " + search("thor"));
    }
}
```

Output:

```
inserted the
inserted a
inserted there
inserted their
inserted any
inserted thee
thee -> true
thor -> false
```

7. Huffman Coding

Huffman coding is a lossless data compression algorithm. The idea is to assign variable-length codes to input characters, lengths of the assigned codes are based on the frequencies of corresponding characters.

String --> “aabcdag” ---> 7 * 2 bytes or
7 * 2 * 8 = 122 bits

size of the “file”

Two HashMap :

encoder

char	String
a b c d	bits

decoder

String	char
bits	a b c d

Steps :-

Step-1:

Pass the String (data, aka(also known as) feeder)

Step-2:

Make a frequency map :

char	Frequency
a	**60**
b	**30**
c	**8**
d	**2**

⟶ a is coming 60 times

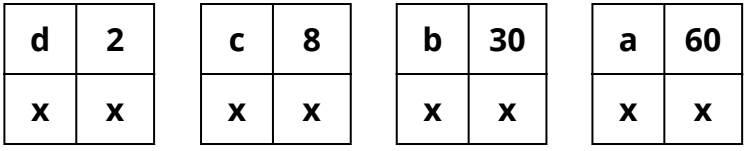
Step-3:

For every key in the frequency map, create a node and insert that node in a min-heap/PriorityQueue.

Node data :

```
char data;  // a
int cost;   // frequency --> 60
Node left;
Node right;
```

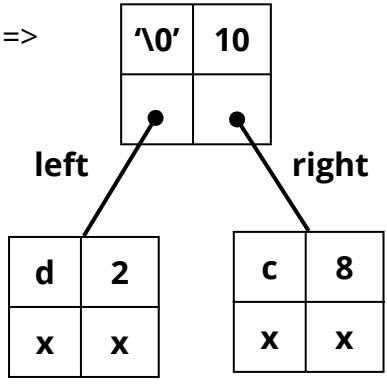
min-heap:



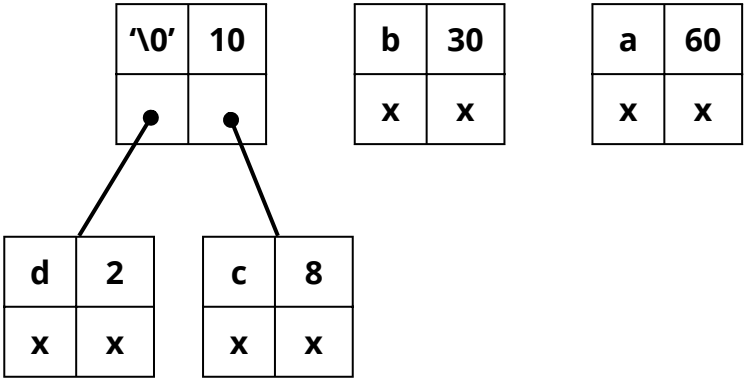
Step-4:

remove 2 elements from min-heap and combine:

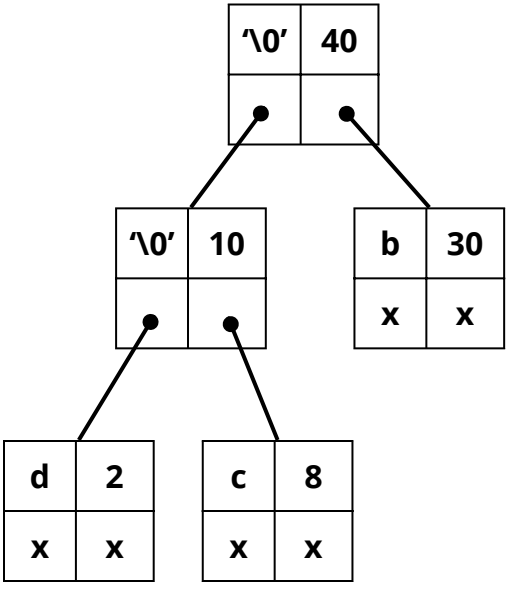
d, c remove =>



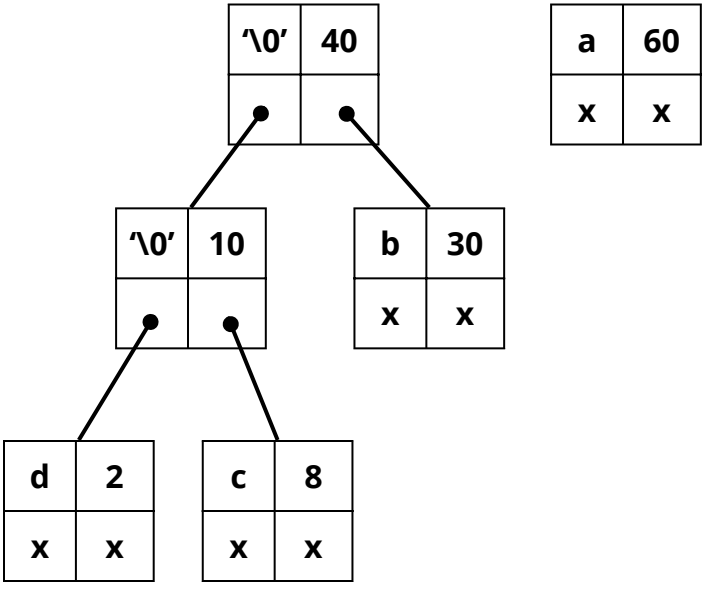
Updated min-heap:



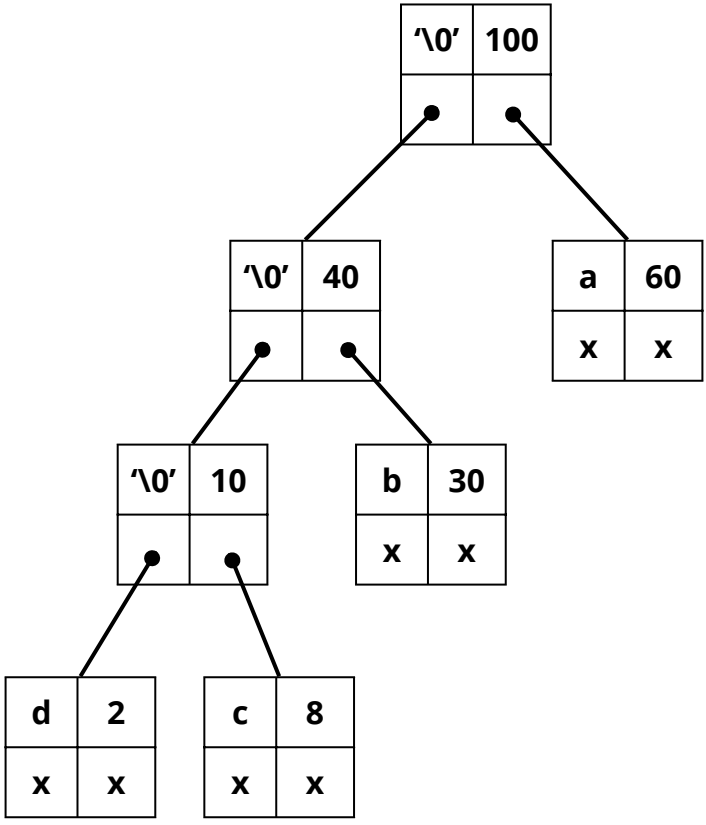
dc, b remove =>



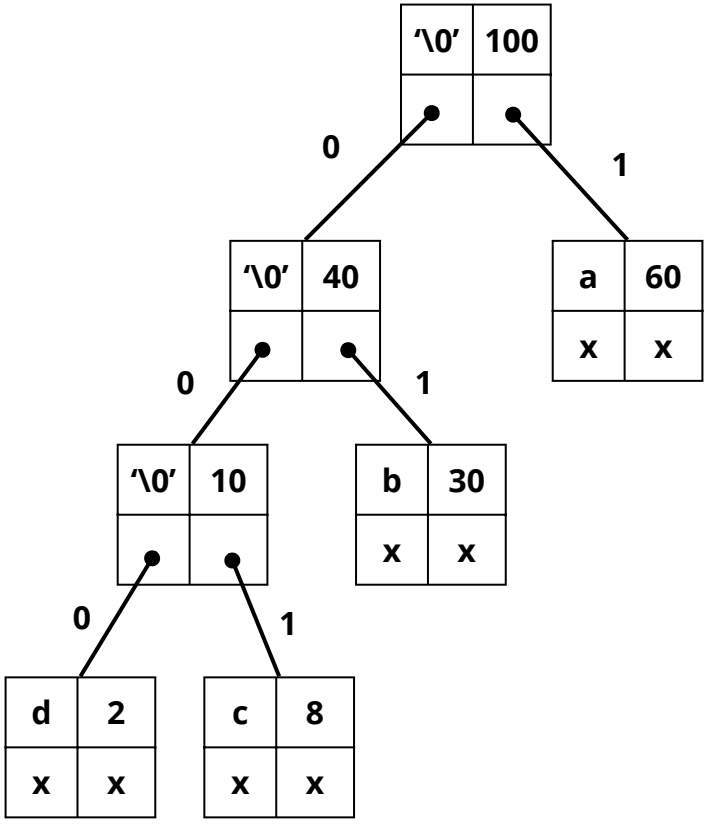
Updated min-heap:



dcb, a remove =>



Updated min-heap:



encoder

char	String
a	1
b	01
c	001
d	000

decoder

String	char
1	a
01	b
001	c
000	d

Step-5:

How to encode / decode :

String = abbccda
= 122 bits (7*2*8)

frequency map:
a=2 ---> a is coming 2 times
b=2
c=2
d=1

After making the frequency tree:-

encoder map:
a = 11
b = 01
c = 10
d = 00

a	b	b	c	c	d	a
11	01	01	10	10	00	11 = 15 bits

Time complexity:
O(nlogn)

where n is the number of unique characters. If there are n nodes, extractMin() is called 2*(n – 1) times. extractMin() takes O(logn) time as it calls minHeapify(). So, the overall complexity is O(nlogn).

If the input array is sorted, there exists a linear time algorithm.

Space complexity:
O(N)

Where, N = Number of nodes

Huffman code implementations:

```
import java.util.*;
import java.util.PriorityQueue;

class HuffmanCoder {
  HashMap<Character, String> encoder;
  HashMap<String, Character> decoder;

  private class Node implements Comparable<Node> {
    Character data;
    int cost; // frequency
    Node left;
    Node right;

    public Node(Character data, int cost) {
      this.data = data;
      this.cost = cost;
      this.left = null;
      this.right = null;
    }

    @Override
    public int compareTo(Node other) {
      return this.cost - other.cost;
    }
  }

  public HuffmanCoder(String feeder) throws Exception {
    HashMap<Character, Integer> fmap = new HashMap<>
();
```

```
    // inserting in frequency map
    for(int i=0; i < feeder.length(); i++) {
      char cc = feeder.charAt(i);
      if(fmap.containsKey(cc)) {
        int ov = fmap.get(cc);
        ov += 1;
        fmap.put(cc, ov);
      } else {
        fmap.put(cc, 1);
      }
    }

    PriorityQueue<Node> minHeap = new PriorityQueue<>
();

    Set<Map.Entry<Character, Integer>> entrySet =
fmap.entrySet();

    // adding all node in minHeap
    for(Map.Entry<Character, Integer> entry : entrySet) {
      Node node = new Node(entry.getKey(),
entry.getValue());
      minHeap.add(node);
    }

    // creating fmap tree
    while(minHeap.size() != 1) {
      Node first = minHeap.remove();
      Node second = minHeap.remove();
```

```
      Node newNode = new Node('\0', first.cost +
second.cost);
      newNode.left = first;
      newNode.right = second;

      minHeap.add(newNode);
    }

    Node ft = minHeap.remove();

    this.encoder = new HashMap<>();
    this.decoder = new HashMap<>();

    this.initEncoderDecoder(ft, "");
  }

  //inserting encoder map and
  //inserting decoder map
  private void initEncoderDecoder(Node node, String osf) {
    if(node == null) {
      return;
    }
    if(node.left == null && node.right == null) {
      this.encoder.put(node.data, osf);
      this.decoder.put(osf, node.data);
    }
    initEncoderDecoder(node.left, osf+"0");
    initEncoderDecoder(node.right, osf+"1");
  }
```

```
  //encoding Strings to bits
  public String encode(String source) {
    String ans = "";

    // Bitset can be used: like an array but with a bit at each
index

    for(int i=0; i<source.length(); i++) {
      ans = ans + encoder.get(source.charAt(i));
    }

    return ans;
  }

  //decoding bits to Strings
  public String decode(String codedString) {
    String key = "";
    String ans = "";
    for(int i=0; i<codedString.length(); i++) {
      key = key + codedString.charAt(i);
      if(decoder.containsKey(key)) {
        ans = ans + decoder.get(key);
        key = "";
      }
    }
    return ans;
  }
}

public class Test {
  public static void main(String[] args) throws Exception{
```

```
    String str = "abbccda";
    HuffmanCoder hf = new HuffmanCoder(str);

    System.out.println("encoder map:");
    System.out.println("a = " + hf.encoder.get('a'));
    System.out.println("b = " + hf.encoder.get('b'));
    System.out.println("c = " + hf.encoder.get('c'));
    System.out.println("d = " + hf.encoder.get('d'));
    String cs = hf.encode(str);
    System.out.println(cs);
    String dc = hf.decode(cs);
    System.out.println(dc);
  }
}
```

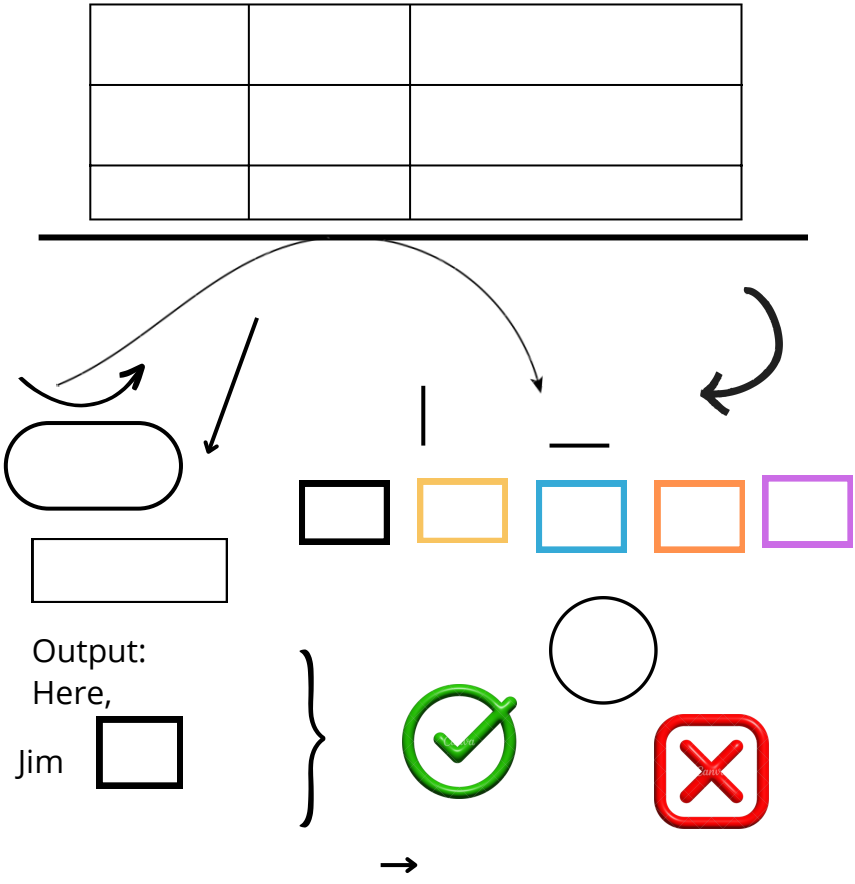
Output:

```
encoder map:
a = 11
b = 01
c = 10
d = 00
11010110100011
abbccda
```

4. ArrayList

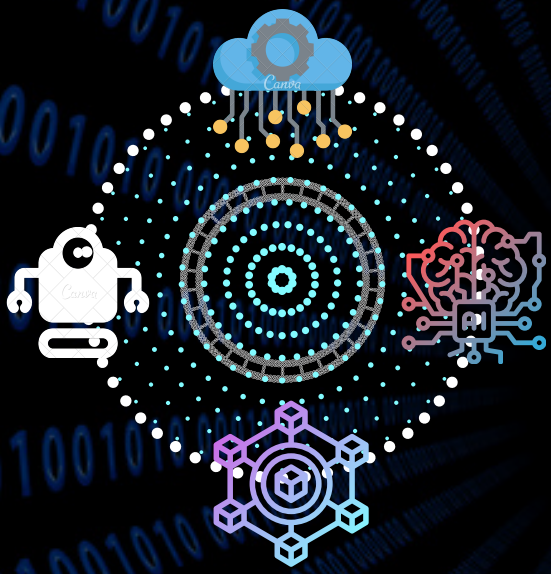
Operator Precedence and Associativity:

- **Make a program that will show String palindrome.**



- **What is the output of the following Java program fragment:**

Operator



JAVA
(THIRD PART)
T.I.M. HAMIM

ABC
PROKASHONI