

ICS1312 – JAVA PROGRAMMING LABORATORY

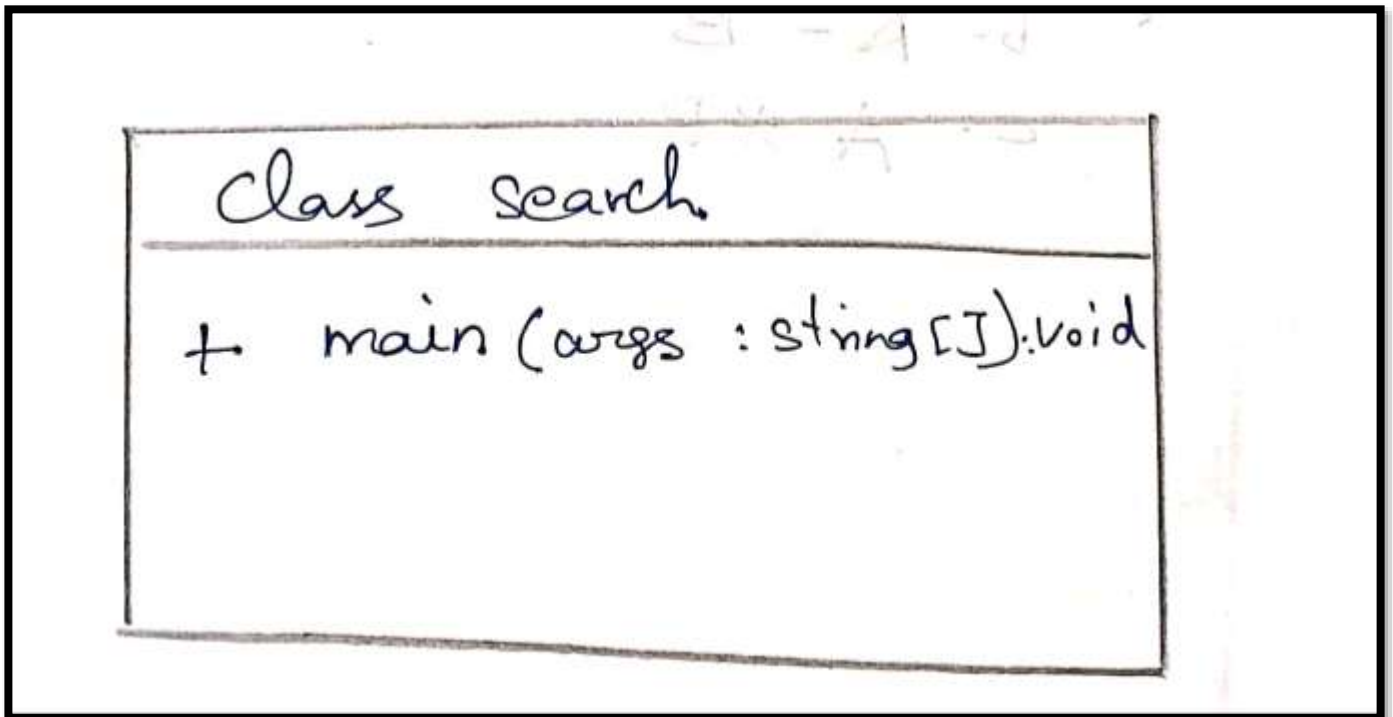
DATE : 24.7.2025
ASSIGNMENT : 1C
TITLE : ARRAYS AND STRINGS
ROLL NO : 3122247001017

LEARNING OBJECTIVE:

- To implement string manipulation tasks in java
- To work with arrays in java
- To perform matrix multiplication, pangram, binary search in java

1. Write a Java program that takes as input 'n' elements, and searches for an element using both linear search and binary search. (CO1, 1.4.1, K3)

CLASS DIAGRAM :



class Diagram:

class search1

+ elements : int []

+ n : int

+ sortelements : data []

+ getelements() : void

+ linear() : int

+ mergesort() : data []

+ merge(L: data [], R: data []) : data []

+ binarysearch()

class data

+ value : int

+ index : int

+ data(value: int, index: int)

CODE:

```
import java.util.Scanner;
class data{
    int value;
    int index;
    data(int value,int index)
    {
        this.index=index;
        this.value=value;
    }
}
class Search1
{
    public int[] elements;
    int n;
    Scanner sc = new Scanner(System.in);
    public void getelements()
    {
        System.out.println("Enter number of elements : ");

        n=sc.nextInt();
        elements = new int[n];
        for(int i=0;i<n;i++)
        {
            System.out.println("Enter element :" + (i+1));
            elements[i]=sc.nextInt();
        }

    }
    public int linear()
    {
        int s;
        boolean status =false;
        System.out.println(" Enter an element to search ");
        s=sc.nextInt();
        for(int i=0;i<n;i++)
        {
            if(elements[i]==s)
```

```

{
    System.out.println("The element found in the index :"+ i);
    status = true;
    return(i);
}

}
if(!status)
    System.out.println("No such element found");
    return(-1);

}
public data[] mergesort(data[] array)
{
    if(array.length <= 1)
    {
        return(array);
    }
    int n=array.length;
    int mid=n/2;

    data[] L=new data[mid];
    data[] R=new data[n-mid];

    for(int i=0;i<mid;i++)
    {
        L[i]=array[i];
    }
    for(int j=0;j<(n-mid);j++)
    {
        R[j]=array[mid+j];
    }

    data[] left=mergesort(L);
    data[] right=mergesort(R);

    return(merge(left,right));

}

```

```

public data[] merge(data[] L,data[] R)
{

    int i=0;
    int j=0;
    int k=0;
    int n=L.length+R.length;
    data[] result=new data[n];

    while(i<L.length && j<R.length)
    {

        if(L[i].value < R[j].value)
        {

            result[k++]=L[i++];
        }
        else
        {

            result[k++]=R[j++];
        }

    }

    while(i<L.length)
    {

        result[k++]=L[i++];
    }

    while(j<R.length)
    {

        result[k++]=R[j++];
    }
    return(result);
}
int count=0;
data[] arr;
data[] sortedelements;
public int binarysearch()
{

```

```

if(count++ == 0)
{
arr=new data[n];
sortedelements=new data[n];
for(int i=0;i<n;i++)
{
arr[i]=new data(elements[i],i);
}
sortedelements=mergesort(arr);
}
int s;
boolean status=false;
System.out.println("Enter an element to search");
s=sc.nextInt();
int low =0;
int high =elements.length-1;
int mid;
while(low <= high)
{
mid = low + (high - low) / 2;
if(sortedelements[mid].value == s)
{
System.out.println("The element found in the index:" +
sortedelements[mid].index);
return(mid);
}
else if(s < sortedelements[mid].value)
{
high=mid-1;
}
else if(s > sortedelements[mid].value)
{
low = mid +1;
}
}
System.out.println("No such element found");
return(-1);
}
}
class Search

```

```
{
public static void main(String[] args)
{
int index;
Scanner sc = new Scanner(System.in);
Search1 s=new Search1();
s.getelements();
while(true)
{
System.out.println("----- MENU -----");
System.out.println("1. PRESS 1 : LINEAR SEARCH");
System.out.println("2. PRESS 2 : BINARY SEARCH");
System.out.println("3. PRESS 3 : EXIT          ");
System.out.println("-----");
int res =sc.nextInt();
if(res == 1)
index=s.linear();
else if(res == 2)
index=s.binarysearch();

else if(res == 3)
break;
else
System.out.println("INVALID INPUT");

}

}

}
```

OUTPUT:

Expected output

Enter no. of element: 5

1
2
3
4
5

Press 1 to linear search

1

Enter element to search: 1

0

Press 2 to Binary search

2

Enter element to search: 4

2

Enter element to search: 9

No such element found

CASE 1:

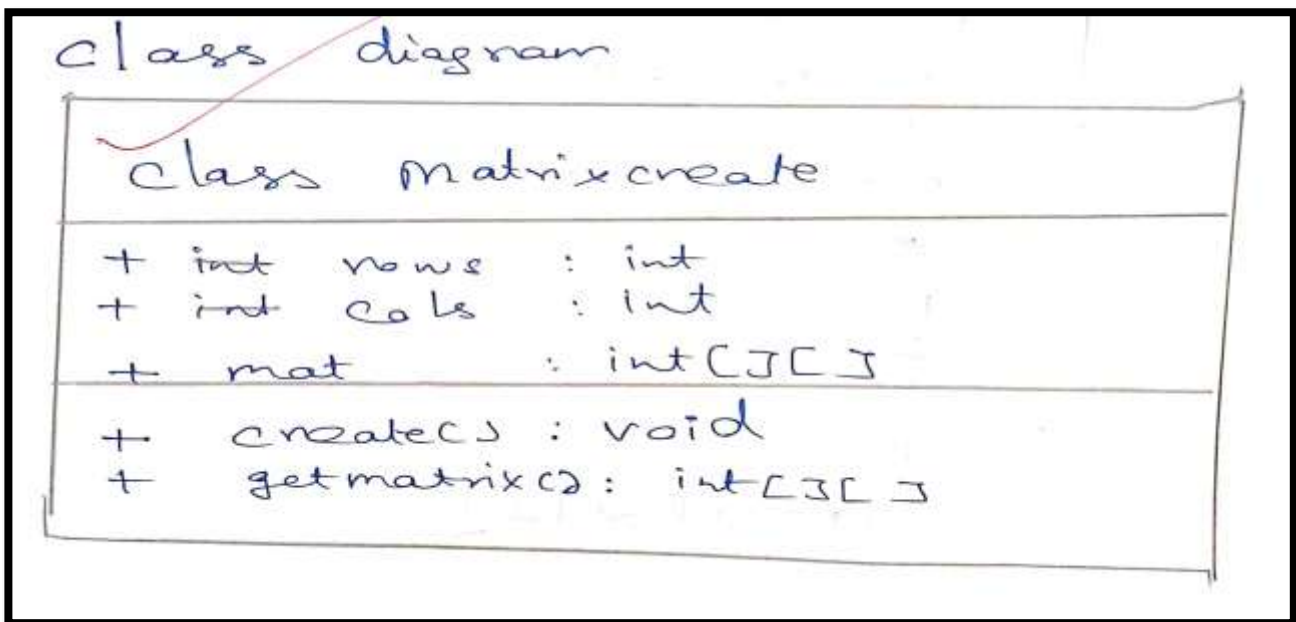
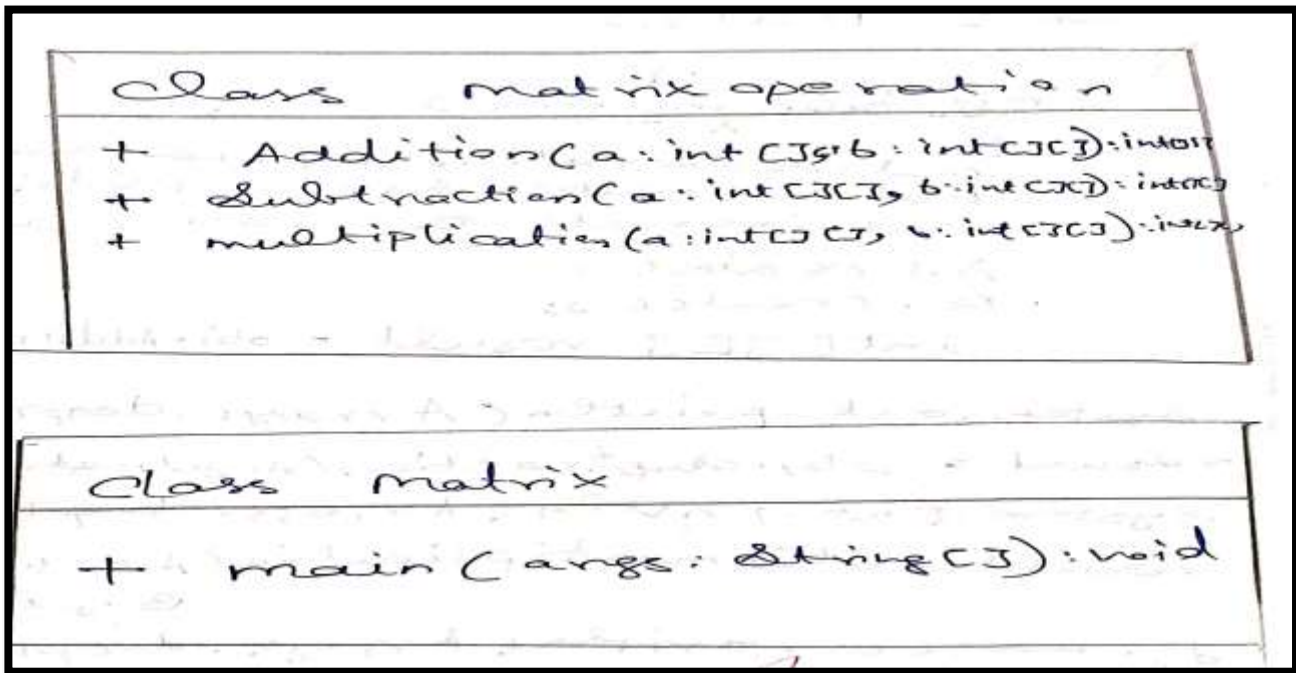
```
The element found in the index :3
----- MENU -----
1. PRESS 1 : LINEAR SEARCH
2. PRESS 2 : BINARY SEARCH
3. PRESS 3 : EXIT
-----
2
Enter an element to search
56
The element found in the index:3
----- MENU -----
1. PRESS 1 : LINEAR SEARCH
2. PRESS 2 : BINARY SEARCH
3. PRESS 3 : EXIT
-----
3
```

ALL POSSIBLE CASSES:


```
Enter number of elements :
6
Enter element :1
89
Enter element :2
78
Enter element :3
67
Enter element :4
56
Enter element :5
45
Enter element :6
34
----- MENU -----
1. PRESS 1 : LINEAR SEARCH
2. PRESS 2 : BINARY SEARCH
3. PRESS 3 : EXIT
-----
1
  Enter an element to search
78
The element found in the index :1
----- MENU -----
1. PRESS 1 : LINEAR SEARCH
2. PRESS 2 : BINARY SEARCH
3. PRESS 3 : EXIT
-----
2
  Enter an element to search
78
The element found in the index:1
----- MENU -----
1. PRESS 1 : LINEAR SEARCH
2. PRESS 2 : BINARY SEARCH
3. PRESS 3 : EXIT
-----
1
  Enter an element to search
56
The element found in the index :3
```

2. Write a Java program that takes as input two $n \times n$ matrices A and B, and displays:
(CO1, 1.4.1, K3)
- $A + B$
 - $A - B$
 - $A \times B$

CLASS DIAGRAM:



CODE:

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

class Matrixcreate
{
    int rows;
    int cols;
    Scanner sc = new Scanner(System.in);
    int[][] mat;
    public void create()
    {

        System.out.println("Enter number of rows :");
        rows=sc.nextInt();
        System.out.println("Enter number of columns :");
        cols=sc.nextInt();
        mat=new int[rows][cols];
        for(int i=0;i<rows;i++)
        {
            for(int j=0;j<cols;j++)
            {

                System.out.println("Enter Element row[" + i + "] column[" + j + "] :");
                mat[i][j]=sc.nextInt();
            }
        }

        System.out.println(Arrays.deepToString(mat));

    }
    public int[][] getmatrix(){
        return mat;
    }
}

class Matrixoperation
{

    public int[][] Addition(int[][] a,int[][] b)
    {
        int row1len =a.length;
        int column1len=a[0].length;

        int row2len=b.length;
        int column2len=b[0].length;
```

```

if(row1len != row2len || column1len != column2len)
{
    System.out.println("The order of the matrix is different : Addition not
possible");
    return(null);
}
int[][] c =new int[row1len][column1len];
for(int i=0;i<row1len;i++)
{
    for(int j=0;j<column1len;j++)
    {
        c[i][j] = a[i][j] + b[i][j];
    }
}
return(c);}

public int[][] subtraction(int[][] a,int[][] b)
{
    int row1len =a.length;
    int column1len=a[0].length;

    int row2len=b.length;
    int column2len=b[0].length;

    if(row1len != row2len || column1len != column2len)
    {

        System.out.println("The order of the matrix is different : Subtraction not
possible");
        return(null);
    }
    int[][] c =new int[row1len][column1len];
    for(int i=0;i<row1len;i++)
    {
        for(int j=0;j<column1len;j++)
        {
            c[i][j] = a[i][j] - b[i][j];
        }
    }
    return(c);
}

```

```

}

public int[][] multiplication(int[][] a,int[][] b)
{
    int row1len =a.length;
    int column1len=a[0].length;
    int row2len=b.length;
    int column2len=b[0].length;
    if(column1len != row2len )
    {
        System.out.println("Multiplication not possible for This  order of the matrix ");
        return(null);
    }
    int[][] c =new int[row1len][column2len];
    for(int i=0;i<row1len;i++)
    {
        for(int j=0;j<column2len;j++)
        {
            for(int k=0;k<row2len;k++)
            {
                c[i][j] += a[i][k]  * b[k][j];
            }
        }
    }
    return(c);
}
}

```

```

class Matrix{

    public static void main(String[] args)
    {
        Matrixcreate A=new Matrixcreate();
        Matrixcreate B=new Matrixcreate();
        System.out.println("Matrix A:");
        A.create();
        System.out.println("Matrix B:");
        B.create();
        Matrixoperation obj=new Matrixoperation();
        int[][] result;
        result=obj.Addition(A.getmatrix(),B.getmatrix());
        System.out.println("Addition result : " + Arrays.deepToString(result));
    }
}

```

```

        result=obj.subtraction(A.getmatrix(),B.getmatrix());
        System.out.println("Subtraction result : " +
Arrays.deepToString(result));
        result=obj.multiplication(A.getmatrix(),B.getmatrix());
        System.out.println("Multiplication result : " +
Arrays.deepToString(result));
    }
}

```

OUTPUT:

Expected output:

Matrix A:

Enter row : (size)

2

Enter Column

2

1

2

3

4

[1, 2], [2, 4]

Matrix B:

Enter number of row :

[1, 2], [3, 4]

2

Enter number of Column

2

1

2

6

8

[1, 2], [6, 8]

Addition result : [2, 4], [7, 12]

Subtraction result : [0, 0], [-3, -4]

Multiplication result : [13, 18], [24, 28]

CASE 1:

```
PS D:\Java\lab2> java Matrix
```

Matrix A:

Enter number of rows :

3

Enter number of columns :

1

Enter Element row[0] column[0] :

1

Enter Element row[1] column[0] :

2

Enter Element row[2] column[0] :

3

[[1], [2], [3]]

Matrix B:

Enter number of rows :

3

Enter number of columns :

1

Enter Element row[0] column[0] :

5

Enter Element row[1] column[0] :

3

Enter Element row[2] column[0] :

2

[[5], [3], [2]]

Addition result : [[6], [5], [5]]

Subtraction result : [[-4], [-1], [1]]

Multiplication not possible for This order of the matrix

Multiplication result : null

CASE 2:

PS D:\Java\lab2> java Matrix

Matrix A:

Enter number of rows :

2

Enter number of columns :

2

Enter Element row[0] column[0] :

1

Enter Element row[0] column[1] :

2

Enter Element row[1] column[0] :

3

Enter Element row[1] column[1] :

4

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

Matrix B:

Enter number of rows :

2

Enter number of columns :

2

Enter Element row[0] column[0] :

5

Enter Element row[0] column[1] :

6

Enter Element row[1] column[0] :

7

Enter Element row[1] column[1] :

8

[[5, 6], [7, 8]]

Addition result : [[6, 8], [10, 12]]

Subtraction result : [[-4, -4], [-4, -4]]

Multiplication result : [[19, 22], [43, 50]]

CASE 3:

PS D:\Java\lab2> java Matrix

Matrix A:

Enter number of rows :

2

Enter number of columns :

3

Enter Element row[0] column[0] :

1

Enter Element row[0] column[1] :

2

Enter Element row[0] column[2] :

3

Enter Element row[1] column[0] :

4

Enter Element row[1] column[1] :

5

Enter Element row[1] column[2] :

6

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

Matrix B:

Enter number of rows :

3

Enter number of columns :

1

Enter Element row[0] column[0] :

1

Enter Element row[1] column[0] :

2

Enter Element row[2] column[0] :

6

[[1], [2], [6]]

The order of the matrix is different : Addition not possible

Addition result : null

The order of the matrix is different : Subtraction not possible

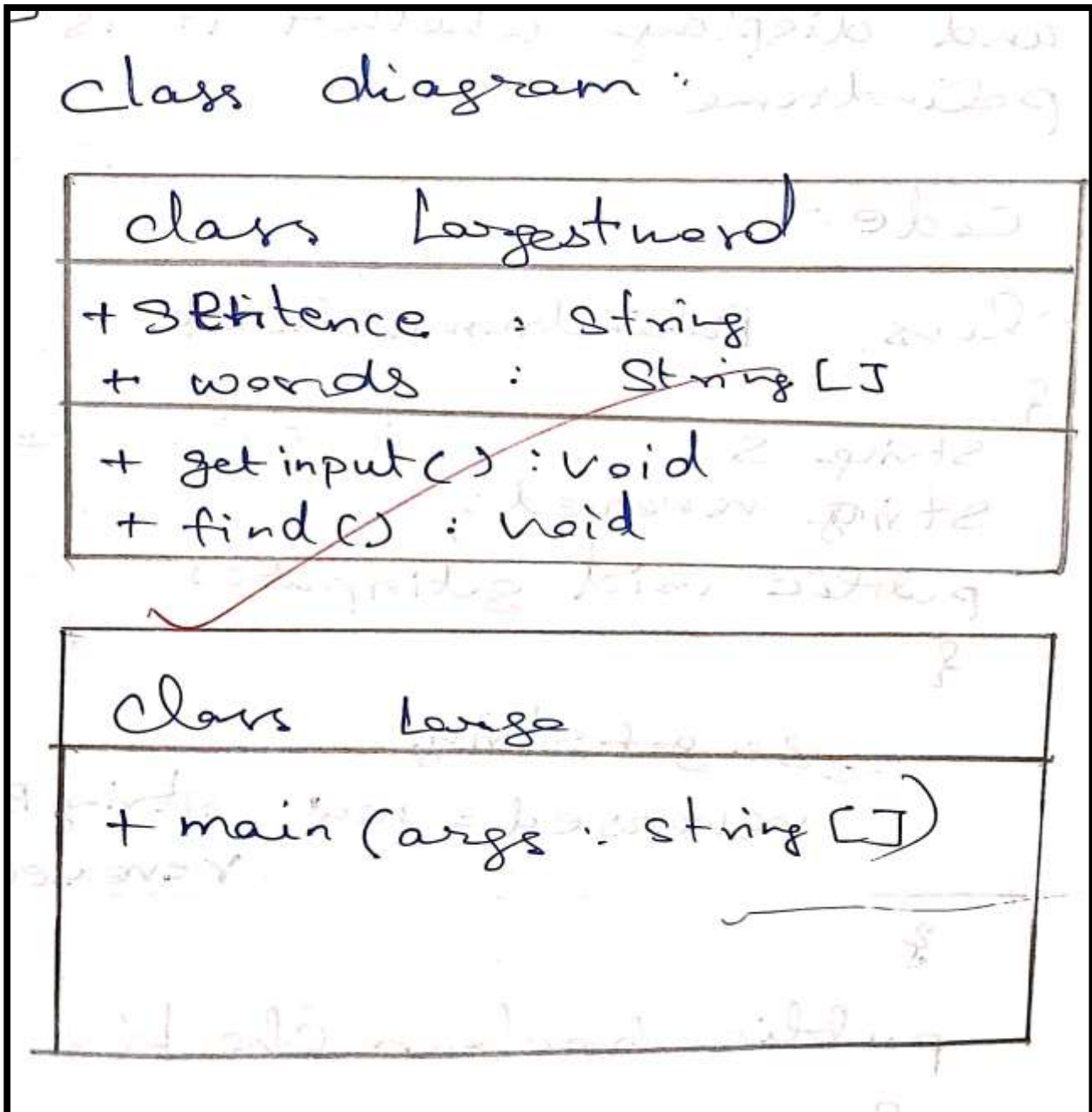
Subtraction result : null

Multiplication result : [[23], [50]]

3. Write a Java program that takes as input an English sentence and displays the longest word in it. Also, print the index position of the first character of that word. (CO1, 1.4.1, K3)

- *Sample Input:* Java is a programming language.
- *Output:* programming, 11

CLASS DIAGRAM :



CODE:

```
import java.util.Scanner;
class Largestword
{

    String sentence;
    String[] words;
    public void getinput()
    {

        System.out.println(" Enter an Sentence : ");
        Scanner sc=new Scanner(System.in);
        sentence=sc.nextLine();
        words=sentence.split(" ");
    }
    public void find()
    {
        int max=words[0].length();
        int index=0;
        int lenplus=0;
        String large=words[0];
        for(String i:words)
        {
            System.out.println(i);
            if(i.length() > max)
            {

                max=i.length();
                large=i;

                index=lenplus;
            }
            lenplus+=i.length()+1;
        }
        System.out.println("The largest word is :" + large + "," + index + " length :"+max);
    }
}
```

```

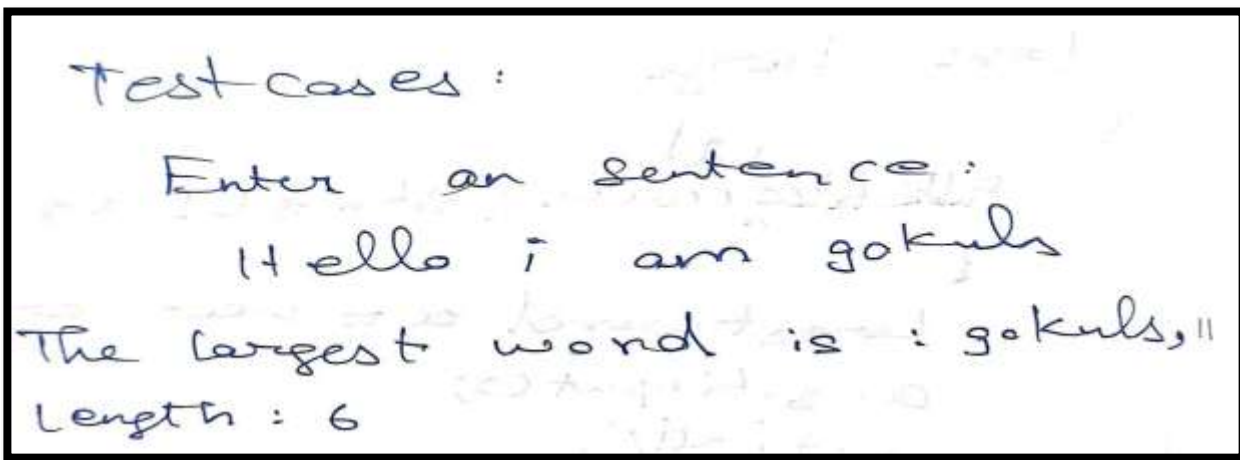
}
class Large{

    public static void main(String[] args){
        Largestword a=new Largestword();
        a.getinput();
        a.find();

    }
}

```

OUTPUT:



Test cases :

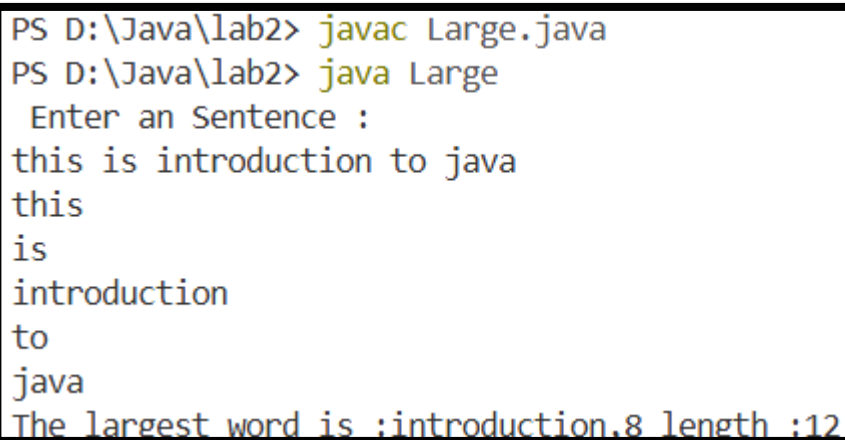
Enter an sentence :

Hello i am gokuls

The largest word is : gokuls, "

Length : 6

CASE 1



```

PS D:\Java\lab2> javac Large.java
PS D:\Java\lab2> java Large
Enter an Sentence :
this is introduction to java
this
is
introduction
to
java
The largest word is :introduction.8 length :12

```

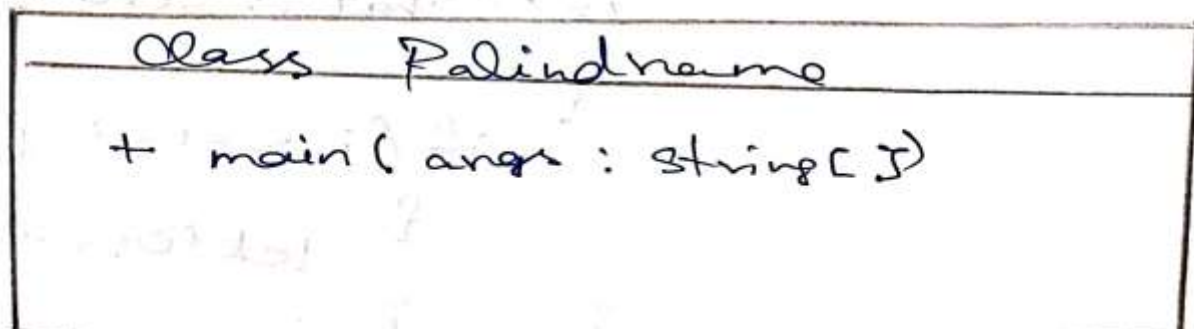
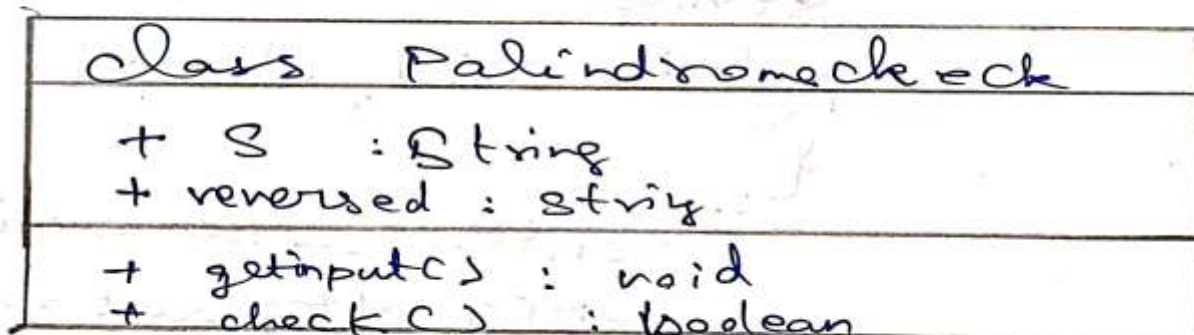
CASE 2

```
PS D:\Java\lab2> java Large
Enter an Sentence :
java is a programming language
java
is
a
programming
language
The largest word is :programming,10 length :11
```

4. Write a Java program that takes as input a string, and displays whether it is a Palindrome or not. (CO1, 1.4.1, K3)

CLASS DIAGRAM:

Class diagram:



CODE:

```
import java.util.Scanner;
class Palindromecheck
{
    String s;
    String reversed;
    public void getinput()
    {
        System.out.println("Enter a string:");
        Scanner sc = new Scanner(System.in);
        s=sc.nextLine();
        reversed=new StringBuilder(s).reverse().toString();
    }
    public boolean check()
    {
        return(reversed.equalsIgnoreCase(s));
    }
}
class Palindrome{
    public static void main(String[] args){

        Palindromecheck a=new Palindromecheck();
        a.getinput();
        boolean result=a.check();
        if(result)
        {
            System.out.println("It is a palindrome");
        }
        else
        {
            System.out.println("It is not a palindrome");
        }
    }
}
```

Test Case :-

Enter a string:

Gokul

It is not a palindrome

Enter a string:

malayalam

It is a palindrome

OUTPUT:

CASE 1:

```
PS D:\Java\lab2> javac Palindrome.java
```

```
PS D:\Java\lab2> java Palindrome
```

```
Enter a string:
```

```
malayalam
```

```
It is a palindrome
```

CASE 2:

```
PS D:\Java\lab2> javac Palindrome.java
```

```
PS D:\Java\lab2> java Palindrome
```

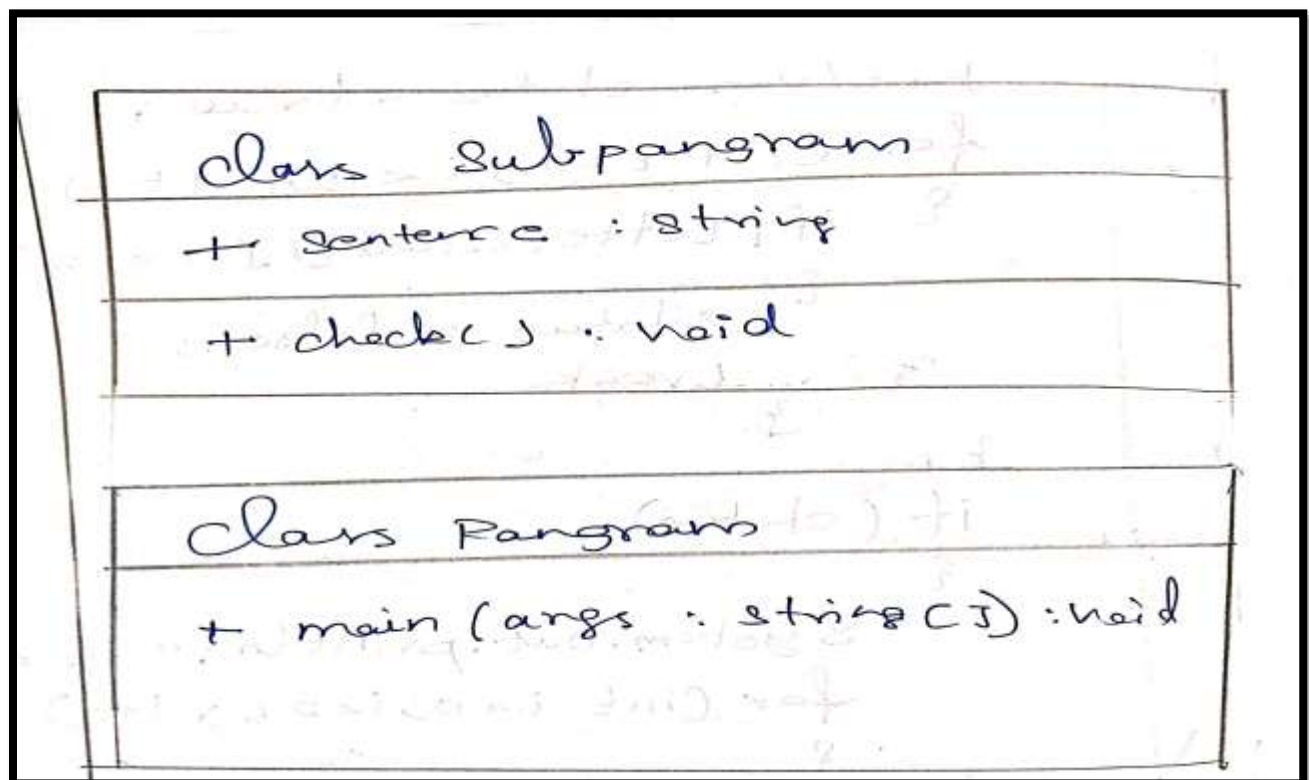
```
Enter a string:
```

```
GOKUL
```

```
It is not a palindrome
```

5. A *Pangram* is a sentence in which all the letters of a given alphabet occur at least once. Write a Java program that displays if a given sentence is a Pangram or not. Also, display the number of occurrences of each letter in the sentence. (CO1, 2.3.1, K3)

CLASS DIAGRAM:



CODE:

```
import java.util.Scanner;
class Palindromecheck
{
    String s;
    String reversed;
    public void getinput()
    {
        System.out.println("Enter a string:");
        Scanner sc = new Scanner(System.in);
        s=sc.nextLine();
        reversed=new StringBuilder(s).reverse().toString();
    }
    public boolean check()
    {
        return(reversed.equalsIgnoreCase(s));
    }
}

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

        Palindromecheck a=new Palindromecheck();
        a.getinput();
        boolean result=a.check();
        if(result)
        {
            System.out.println("It is a palindrome");
        }
        else
        {
            System.out.println("It is not a palindrome");
        }
    }
}
```

TEST CASES:

Test cases

Enter a sentence:

The quick brown fox jumps
the lazy dog

yes ✓

a: 1

b: 1

c: 1

d: 1

e: 3

f: 1

g: 1

h: 2

i: 1

j : 1

k : 1

l : 1

m : 1

n : 1

o : 4

p : 1

q : 1

r : 2

s : 1

t : 1

u : 2

v : 1

w : 1

x : 1

y : 1

z : 1

OUTPUT:

CASE 1:

```
PS D:\Java\lab2> javac Pangram.java
```

```
PS D:\Java\lab2> java Pangram
```

Enter an sentence :

the quick brown fox jumps over the lazy dog

The given string is a panagram

97 : 1

98 : 1

99 : 1

100 : 1

101 : 3

102 : 1

103 : 1

104 : 2

105 : 1

106 : 1

107 : 1

108 : 1

109 : 1

110 : 1

111 : 4

112 : 1

113 : 1

114 : 2

115 : 1

116 : 2

117 : 2

118 : 1

119 : 1

120 : 1

121 : 1

122 : 1

CASE 2:

```
PS D:\Java\lab2> java Pangram
Enter an sentence :
JAVA IS A PROGRAMMING LANGUAGE
The given String is not a panagram
```

LEARNING OUTCOMES:

- Learned to implement string manipulation tasks in java
- Learned to work with arrays in java
- Learned to define logic for merge sort in java