Java lab 2

Lab Exercises

lab1:
Split a string represent an IP address: ➢ Use string split("\\.") ➢ Use StringTokenizer ➢ The program, for example: your Input : ➢ 163.121.12.30 ➢ The Result is : 163 121 12 30

Open notepad write this:

```
import java.util.StringTokenizer;
class IpCutter{
	String cmdLine;

	public IpCutter(String _cmdLine)
	{
		cmdLine=_cmdLine;
	}

	int[] doIpSplitStringClass(){
		String [] ipPartsStr =  cmdLine.split("\\.");
```

```
            int [] ipParts =  new int [ipPartsStr.length];
            int index = 0;
            for (String part : ipPartsStr) {
            ipParts[index] = Integer.parseInt(part);
                    index++;
        }
            return ipParts;
        }

        int[] doIpSplitStringTokenizer(){
            int index=0;
            StringTokenizer st = new StringTokenizer(cmdLine,".");
            int[] ipParts = new int[st.countTokens()];
            while (st.hasMoreTokens()) {
                 ipParts[index]= Integer.parseInt(st.nextToken());
                 index++;
            }
            return ipParts;
        }
}

public class LectureDemo{
        public static void main(String [] args){
            System.out.println("Nada Mohamed Ahmed");
            if(args.length != 0)
            {
                    IpCutter cut = new IpCutter(args[0]);

                    System.out.println("The output of "+ args[0]+" using
StringTokenizer is");
                    int[] stringTokenizerOut =
cut.doIpSplitStringTokenizer();
                    for(int i = 0 ; i < stringTokenizerOut.length ; i++ )
                    {
                            System.out.println(stringTokenizerOut[i]);

                    }
```

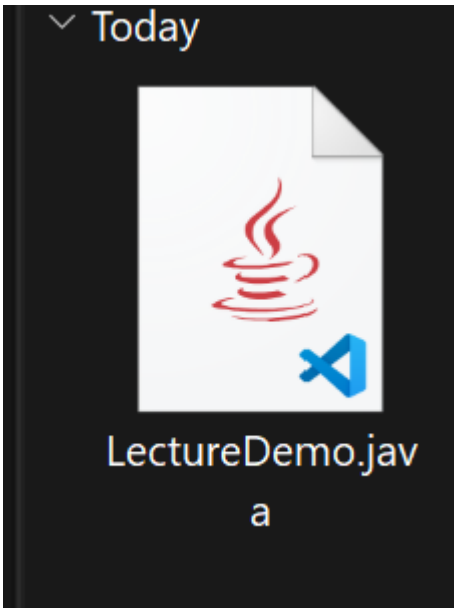
```
                    System.out.println("----------------------------------------");

                    System.out.println("The output of "+ args[0]+" using
String methods is");
                    int[] stringOut = cut.doIpSplitStringClass();
                    for(int i = 0 ; i < stringOut.length ; i++ )
                    {
                            System.out.println(stringOut[i]);

                    }

            }
      }

}
```

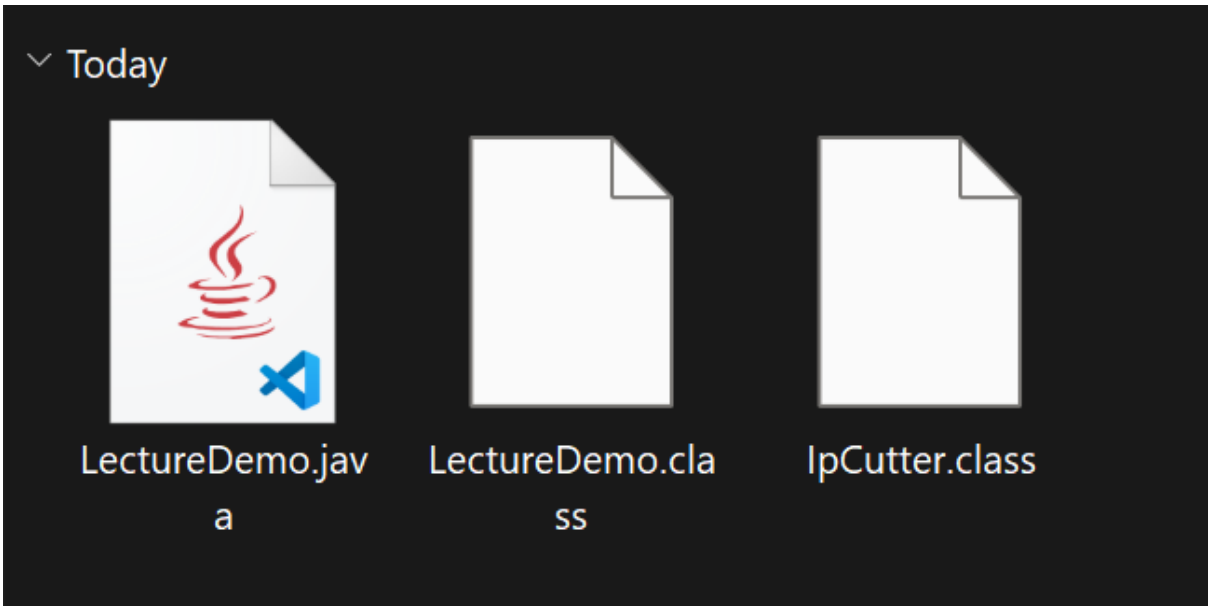


Make the file name :
public class name.java

Then on CMD :
Compile the file :
Javac LectureDemo.java

→ the byte code (intermediate code) appears LectureDemo.class and Ipcutter.class (we will have number of intermediate code file depend on the number of classes)

Run the file :
Java LectureDemo args
→ intermediate code file which contain the main function




```
C:\Users\nadam\Downloads\open source\Java\day2\lab\lab2.1>javaC LectureDemo.java
```

```
C:\Users\nadam\Downloads\open source\Java\day2\lab\lab2.1>java LectureDemo 163.121.12.30
Nada Mohamed Ahmed
The output of 163.121.12.30 using StringTokenizer is
163
121
12
30
----------------------------------------
The output of 163.121.12.30 using String methods is
163
121
12
30

C:\Users\nadam\Downloads\open source\Java\day2\lab\lab2.1>
```

We use :

countTokens()
Calculates the number of times that this tokenizer's nextToken method can be called before it generates an exception.

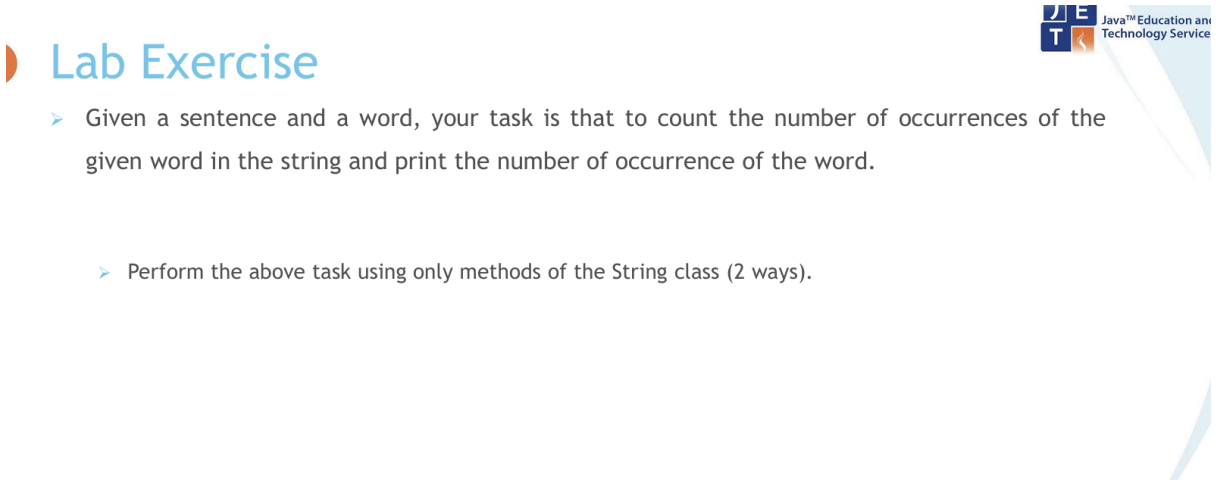
nextToken()
Returns the next token from this string tokenizer.

hasMoreTokens()
Tests if there are more tokens available from this tokenizer's string.

String.split() take regex (a pattern used to match sequences of characters within a string)

Lab2:

Given a sentence and a word, your task is that to count the number of occurrences of the given word in the string and print the number of occurrence of the word. ➢ Perform the above task using only methods of the String class (2 ways).



Same steps

```
class WordProcessing {
        String sentence;
        String word;

        public WordProcessing(String _sentence, String _word) {
        sentence = _sentence;
        word = _word;
        }

        int WordCountUsingSplit() {
        int countOccurrence = 0;
        String[] parts = sentence.split("[\\s.,]+");

        for (String part : parts) {
                if (part.equals(word)) {
                countOccurrence++;
                }
        }
        return countOccurrence;
        }

        int WordCountUsingIndexOf() {
        int countOccurrence = 0;
                int index = sentence.indexOf(word);
```

```
		while(index != -1)
		{
			countOccurrence++;
			index = sentence.indexOf(word , index+1);

		}

	return countOccurrence;
	}


}

public class LectureDemo2 {
	public static void main(String[] args) {
	if (args.length < 2) {
		System.out.println("Please enter sentence and word.");
	}
		else{
			String sentence = args[0];
		String word = args[1];

		WordProcessing process = new WordProcessing(sentence, word);
		int numOfOccurrenceSplit = process.WordCountUsingSplit();
		System.out.println("The number of occurrences of the word " + word + " in the
sentence " + sentence + " is: " + numOfOccurrenceSplit);

			int numOfOccurrenceIndexOf = process.WordCountUsingIndexOf();
		System.out.println("The number of occurrences of the word " + word + " in the
sentence " + sentence + " is: " + numOfOccurrenceIndexOf);
	}}
}
```

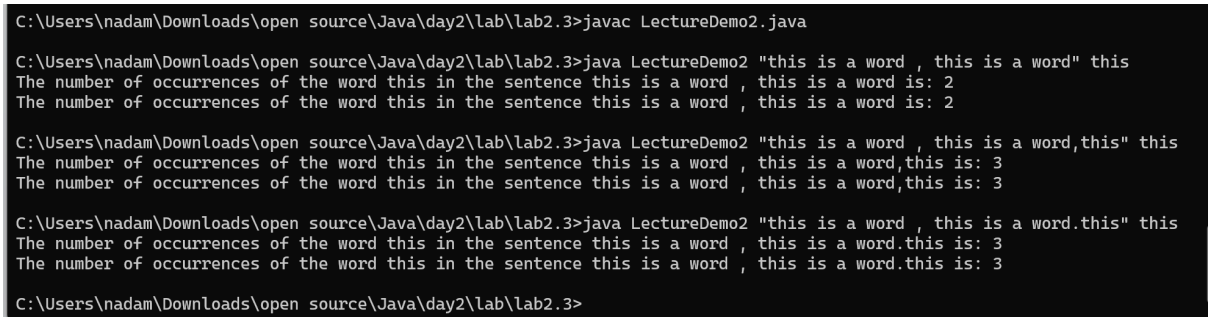
We use :-

`indexOf(int ch)`

Returns the index within this string of the first occurrence of the specified character.

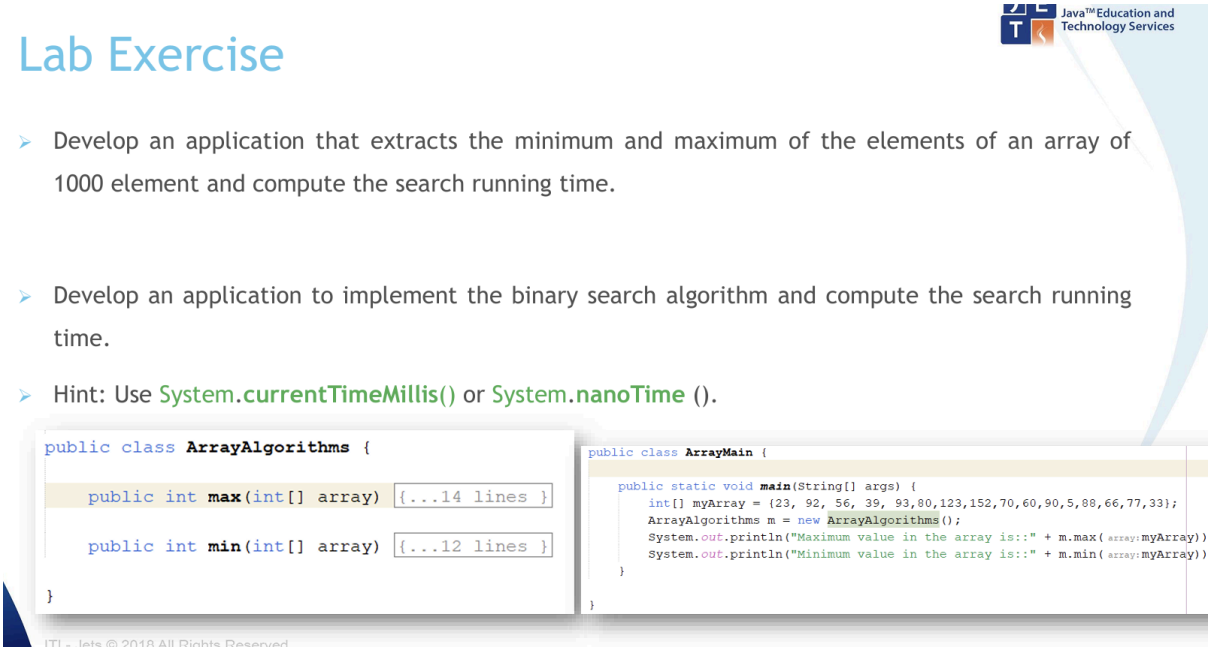
`indexOf(int ch, int fromIndex)`

Returns the index within this string of the first occurrence of the specified character, starting the search at the specified index.



Lab3 :

Develop an application that extracts the minimum and maximum of the elements of an array of 1000 elements and computes the search running time. ➢ Develop an application to implement the binary search algorithm and compute the search running time. ➢ Hint: Use System.currentTimeMillis() or System.nanoTime ()




```
import java.util.Random;
import java.util.Arrays;

class ArrayProcessing {
	int findMin(int [] arr)
	{
		long start = System.nanoTime();
		int min = arr[0];
		for(int i = 1 ; i< arr.length ; i++)
		{
			if(min > arr[i])
			{
				min = arr[i];
			}
```

```
        }
        long end = System.nanoTime();
        long duration = end - start;
        System.out.println("the duration of this process = "+duration);
        return min;
}
int findMax(int [] arr)
{
        long start = System.nanoTime();

        int max = arr[0];
        for(int i = 1 ; i< arr.length ; i++)
        {
                if(max < arr[i])
                {
                        max = arr[i];
                }
        }
        long end = System.nanoTime();
        long duration = end - start;
        System.out.println("the duration of this process = "+duration);

        return max;
}

int binarySearch(int [] arr,int key)
{
        long start = System.nanoTime();

        int low = 0;
      int high = arr.length - 1;
        int mid;

        while(low <= high){
        mid = (low+high)/2;
                if(key == arr[mid]){
                        long end = System.nanoTime();
```

```
                        long duration = end - start;
                        System.out.println("the duration of this process =
"+duration);
                        return mid;
                }
                        else if(key > arr[mid]){
                                low = mid+1;
                }
                else{
                                high = mid-1;
                }
                }
                return -1;
        }


}

public class LectureDemo3 {
        public static void main(String[] args) {
                System.out.println("Nada");

                Random rand = new Random();
                int[] array = new int[1000];
                for(int i=0 ; i<1000;i++)
                {
                        array[i]= rand.nextInt(100);
                }

                ArrayProcessing arr = new ArrayProcessing();
                int maxValue = arr.findMax(array);
                System.out.println("The Max value of the array = "+
maxValue);

                int minValue = arr.findMin(array);
                System.out.println("The Min value of the array = "+
minValue);
```

```
            Arrays.sort(array);
            int key = 6;
            int result = arr.binarySearch(array, key);

        if (result != -1)
                    System.out.println("our key " + key +" found at index: "
+ result);
        else
                    System.out.println("key not found in the array");

      }
}
```

```
C:\Users\nadam\Downloads\open source\Java\day2\lab\lab2.4>javac LectureDemo3.java

C:\Users\nadam\Downloads\open source\Java\day2\lab\lab2.4>java LectureDemo3
Nada
the duration of this process = 11600
The Max value of the array = 99
the duration of this process = 13400
The Min value of the array = 0
the duration of this process = 1200
our key 6 found at index: 53

C:\Users\nadam\Downloads\open source\Java\day2\lab\lab2.4>
```

We use :
Random class
→ import java.util.Random;
→ **`nextInt`**`(int bound)`
Returns a pseudorandom, uniformly distributed `int` value between 0 (inclusive) and the specified value (exclusive), drawn from this random number generator's sequence.