**PRACTICAL-6**

**6.1**

**AIM:**  Write a program that will count the number of lines in each file that is specified on the command line. Assume that the files are text files. Note that multiple files can be specified, as in "java Line Counts file1.txt file2.txt file3.txt". Write each file name, along with the number of lines in that file, to standard output. If an error occurs while trying to read from one of the files, you should print an error message for that file, but you should still process all the remaining files.

**PROGRAM:**

import java.io.\*;

import java.util.\*;

class Main{

public static void main(String args[]){

try{

File ob1 = new File("Xandu.txt");

ob1.createNewFile();

Scanner myReader1 = new Scanner(ob1);

int count1 =0;

while(myReader1.hasNextLine())

{

myReader1.nextLine();

count1++;

}

System.out.println("file name and number of lines are " +ob1.getName() +" " + count1);

myReader1.close();

}catch(Exception e)

{

System.out.println(e);

}

}

}

**OUTPUT:**

****

**CONCLUSION:** We came to know how to write data in a file using file i/o operation.

**6.2**

**AIM:** Write an example that counts the number of times a particular character, such as e, appears in a file. The character can be specified at the command line. You can use xanadu.txt as the input file.

**PROGRAM:**

import java.io.\*;

import java.util.\*;

class s6\_p2

{

public static void main(String args[])

{

try

{

File ob1 = new File("t1.txt");

ob1.createNewFile();

FileReader Reader = new FileReader(ob1);

if(ob1.exists())

{

int i,count=0;

//ob1.createNewFile();

while((i = Reader.read()) != -1)

{

char ch= (char)i;

if(ch==args[0].charAt(0))

{

count++;

}

}

System.out.println("The number of times " + args[0] + " occurs is: " + count);

}

else

{

System.out.println("File does not exist...");

}

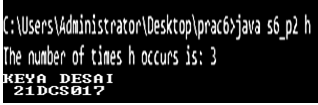
}

catch(Exception e)

{

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

**OUTPUT:**

****

**CONCLUSION:** In this practical, we have seen how to read through a file.

**6.3**

**AIM:**Write a Java Program to Search for a given word in a File. Also show use of Wrapper Class with an example.

**PROGRAM:**

import java.io.\*;

import java.util.\*;

class s6\_p3

{

public static void main(String args[])

{

try

{

File ob1 = new File("t1.txt");

//ob1.createNewFile();

FileReader Reader = new FileReader(ob1);

BufferedReader br =new BufferedReader(Reader);

int i,count=0;

String s,s1[];

if(ob1.exists())

{

while((s = br.readLine())!= null)

{

s1=s.split(" ");

for(String s2:s1)

{

if(s2.equals(args[0]))

{

count++;

}

}

}

System.out.println(count);

}

}

catch(Exception e)

{

System.out.println(e);

}

}

}

**OUTPUT:**

****

**CONCLUSION:** In this practical, we have seen how to find a word in file.

**6.4**

**AIM:**Write a program to copy data from one file to another file. If the destination file does not exist, it is created automatically.

**PROGRAM:**

* **Using FileReader/FileWriter class:**

import java.io.\*;

import java.util.\*;

//fileinputstream

class s6\_p4\_1

{

public static void main(String args[])

{

try

{

FileReader fr = new FileReader ("t1.txt");

FileWriter fw = new FileWriter ("t2.txt");

int c;

while((c = fr.read()) != -1)

{

fw.write(c);

}

System.out.println("File copied successfully");

fr.close();

fw.close();

}

catch(Exception e)

{

System.out.println(e);

}

}

}

**OUTPUT:**

****

**Conclusion:**

In this practical we learn the concept of file i/o that counts number of lines.

**6.5:**

**AIM:**Write a program to show use of character and byte stream. Also show use of BufferedReader/BufferedWriter to read console input and write them into a file.

**Character stream class:**

* They are used to perform input and output for 16-bit Unicode.

**PROGRAM:**

import java.io.\*;

import java.util.\*;

class file

{

public static void main(String args[])

{

try

{

File ob1 = new File("t1.txt");

ob1.createNewFile();

FileWriter fw = new FileWriter(ob1);

String s;

Scanner sc = new Scanner(System.in);

System.out.println("Please enter content:");

s = sc.nextLine();

for(int i =0 ; i<(s.length()) ; i++)

{

fw.write(s.charAt(i));

}

fw.close();

}

catch(Exception e)

{

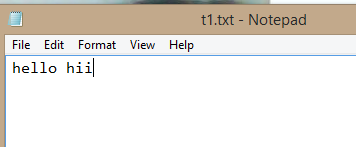
System.out.println(e);

}

}

}

**OUTPUT:**



**Conclusion:**From this practical we understand that **In java, file handling is nothing but a process of reading the data from a file and writing the data to a file.**

**PRACTICAL-7**

**7.1**

**AIM:** Write a program to create thread which display “Hello World” message. A. by extending Thread class B. by using Runnable interface.

**Program:**

class Mythread extends Thread{ public void run(){

System.out.println("hello"); System.out.println("hii"); System.out.println("hello world");

}

}

public class Main{

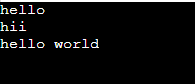
public static void main(String[] args){ Mythread m = new Mythread(); m.start();

System.out.println("Keya Desai 21DCS017");

}

}

**Output :**

****

**Conclusion :**

In this Practical , we can learn about basic of multithreading.

**7.2**

**AIM:** WriteaprogramwhichtakesNandnumberofthreadsasanargument.Programshoulddistribute the task of summation of N numbers amongst number of threads and final result to be displayed on theconsole.

**Program:**

import java.util.\*;

class Main{

public static void main(String args[]){ mythread i = new mythread(); i.start();

}

}

class mythread extends Thread { int n;

int t,S=0;

int[] T = new int[20]; public void run(){

Scanner sc = new Scanner(System.in); System.out.println("Enter your Number"); int n = sc.nextInt();

System.out.println("Enter number of thread"); int t = sc.nextInt();

int p =n;

for(int i=1;i<=t;i++){

if(i != t){

T[i] = n/t;

System.out.println("Thred "+i+"is equal to :" +T[i]); n=n-T[i];

}

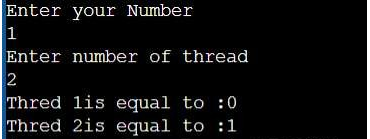
}

for(int i=1; i<= (t-1) ; i++){ S = S + T[i];

}

System.out.println("Thred "+t+"is equal to :"+ (p-S));

**Output :**

****

**Conclusion :**

In this Practical we can learn about how multithreading works and we divide number in some parts

**7.3**

**AIM:** Write a java program that implements a multi-thread application that has three threadsFirstthreadgeneratesrandom integerevery1secondandifthevalueiseven,secondthreadcomputes the square of the number and prints. If the value is odd, the third thread will print the value of cube of thenumber.

**Program:**

public void run(){

System.out.println("Square of the number is:"+(snum\*snum));

}

}

class cube extends Thread{ int cnum;

cube(int number){ cnum=number;

}

public voidrun(){

System.out.println("Cube of the number is:"+(cnum\*cnum\*cnum));

}

}

public class Main{

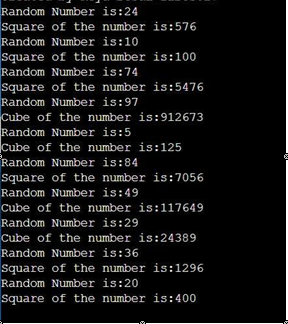
public static void main(String args[]){ number n=new number();

n.start();

}

}

**Output :**

****

**Conclusion :**

In this practical we learn about how we can sleep thread and how to take random numbers in thread .

**7.4**

**AIM:** Write a program to increment the value of one variable by one and display it after one second using thread using sleep() method.

**Program:**

public class Main extends Thread{ int num=10;

@Override

public void run(){ System.out.println(""); for(int i=0; i<=num;i++){

System.out.println(i); try{

Thread.sleep(2000); //Milliseconds

}

catch(InterruptedException e){ e.getMessage();

}

}

}

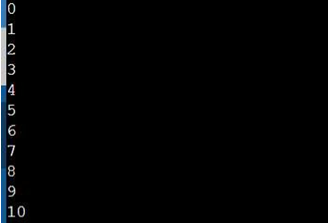
public static void main(String[] args){ Main ob= new Main();

ob.start();

}

}

**Output :**

****

**Conclusion :**

In this practical we can learn about **OVERRIDE** methods .

**7.5**

**AIM:**

Write a program to create three threads ‘FIRST’, ‘SECOND’, ‘THIRD’. Set the priority of the ‘FIRST’ thread to 3, the ‘SECOND’ thread to 5(default) and the ‘THIRD’ thread to 7.

**Program:**

class first extends Thread{ public void run(){

System.out.println(" First thread priority: "+getPriority());

}

}

class second extends Thread{ public void run(){

System.out.println(" Second thread priority: "+getPriority());

}

}

class third extends Thread{ public void run(){

System.out.println(" Third thread priority: "+getPriority());

}

}

class Main{

public static void main(String[] args){ first f= new first();

second s= new second(); third t = new third(); f.setPriority(3); s.setPriority(5); t.setPriority(7);

f.start();

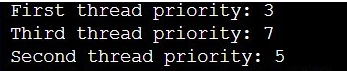
s.start();

t.start();

}

}

**Output :**

****

**Conclusion :**

In this practical we can learn about thread priority.

**7.6**

**AIM:** Write a program to solve producer-consumer problem using thread synchronization.

**Program:**

import java.util.ArrayList; class Main{

public static void main(String[] args) throws InterruptedException{ ProducerConsumer p1 = new ProducerConsumer();

Thread myThread1 = new Thread(new Runnable() { public void run() {

try {

p1.produce();

}catch (InterruptedException e) { e.printStackTrace();

}

}

});

Thread myThread2 = new Thread(new Runnable() { public void run(){

try{

p1.consume();

}catch (InterruptedException e){ e.printStackTrace();

}

}

});

myThread1.start(); myThread2.start(); myThread1.join(); myThread2.join();

}

public static class ProducerConsumer{ ArrayList<Character> list = new ArrayList<Character>(); int capacity = 2;

public void produce() throws InterruptedException { char value = 'a';

while (true){ synchronized (this){

while (list.size() == capacity) wait();

System.out.println("Producer produced product :" + value); list.add(value);

value++;

if (value == 'g'){ System.exit(0); break;

}

notify(); Thread.sleep(1000);

}

}

}

public void consume() throws InterruptedException{ while (true){

synchronized (this){ while (list.size() == 0) wait();

char val = list.remove(0); if (val == 'g'){

break;

}

System.out.println("Consumer consumed Product :" +val); notify();

Thread.sleep(1000);

}

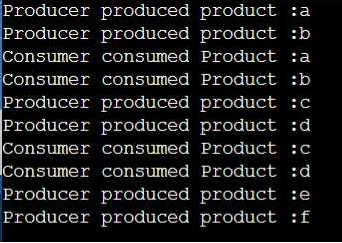
}

}

}

}

**Output :**

****

**Conclusion :**

In this practical we solve about Producer – consumer problem .

**PRACTICAL-8**

**8.1**

**AIM:** Design a Custom Stack using ArrayList class, which implements following functionalities of stack. My Stack

-list ArrayList&lt;Object&gt;: A list to storeelements.

+isEmpty: boolean: Returns true if this stack isempty.

+getSize(): int: Returns number of elements in this stack.

+peek(): Object: Returns top element in this stack without removing it.

+pop(): Object: Returns and Removes the top elements in this stack.

+push(o: object): Adds new element to the top of this stack.

**Program:**

import java.util.\*;

class Mystack{ ArrayList<Object> l; String s;

Mystack(Object element[]){

l = new ArrayList<Object>(); for(int i=0; i<element.length;i++){

l.add(element[i]);

}

}

boolean isEmpty(){ return l.isEmpty();

}

int getsize(){ return l.size();

}

Object peek(){

return l.get(l.size()-1);

}

Object pop(){

return l.remove(l.size()-1);

}

void push(Object o){ l.add(0);

}

void print(){

for(int i=0;i<l.size();i++){ System.out.println(l.get(i));

}

}

}

public class Main{

public static void main(String args[]){ Integer arr[] = newInteger[]{1,2,3}; Mystack s = new Mystack(arr); s.push(7);

s.push(5);

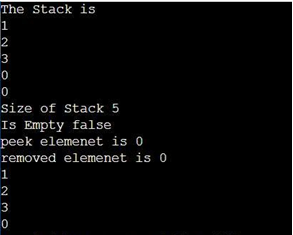
System.out.println("The Stack is "); s.print();

System.out.println("Size of Stack "+s.getsize()); System.out.println("Is Empty "+s.isEmpty()); System.out.println("peek elemenet is "+s.peek()); System.out.println("removed elemenet is "+s.pop()); s.print();

}

}

**Output :**

****

**Conclusion :**

In this practicle, we can learn about ArrayList , isEmpty , getSize() , Peek() , Pop() and push() methods

**8.2**

**AIM:**

Create a generic method for sorting an array of Comparable objects.

**Program:**

import java.util.\*;

class Student implements Comparable<Student>{ String Name;

int javaMarks; Student(){

Name=null; javaMarks=0;

}

Student(String name,int javaMarks){ this.Name=name; this.javaMarks=javaMarks;

}

public int compareTo(Student s){ return this.javaMarks-s.javaMarks;

}

public String toString(){

return String.format("[%s,%d]",Name,javaMarks);

}

}

class Main{

public static void main(String args[]){ Student[] s=new Student[5]; s[0]=new Student("Isha",78); s[1]=new Student("Meera",88); s[2]=new Student("Nirali",56); s[3]=new Student("Abhishek",100); s[4]=new Student("Umang",66);

System.out.println("Before Sorting:"+Arrays.toString(s)); System.out.println("");

Arrays.sort(s);

System.out.println("After Sorting:"+Arrays.toString(s));

**Output :**

****

**Conclusion :**

In this Practical , we can learn about Short And Array to String class.

**8.3**

**AIM:** Write a program that counts the occurrences of words in a text and displays the words and their occurrences in alphabetical order of the words. Using Map and Set Classes.

**Program:**

import java.util.\*;

class Main{

public static void main(String args[]){ Map<String,Integer> map = new HashMap<>(); Scanner Sc = new Scanner(System.in); System.out.println("Enter the String :");

String sentence = Sc.nextLine();

String word = sentence.toLowerCase(); String[] tokens = word.split(" "); for(String token : tokens){

if(map.containsKey(word)){ int count = map.get(word); map.put(word, count+1);

}

else{

map.put(word,1);

}

}

Set<String> keys = map.keySet();

TreeSet<String> sortedKeys = new TreeSet<>(keys); for(String str : sortedKeys){

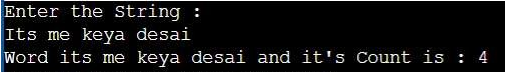
System.out.println("Word "+str+" and it's Count is : "+map.get(str));

}

}

}

**Output :**

****

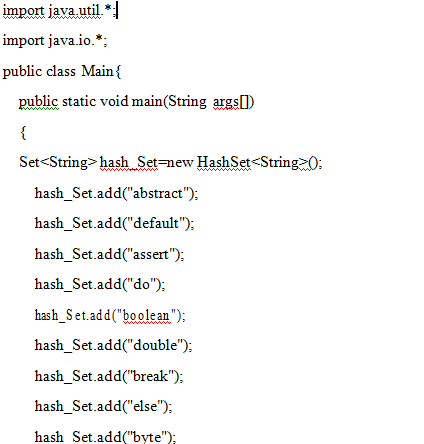
**Conclusion :**

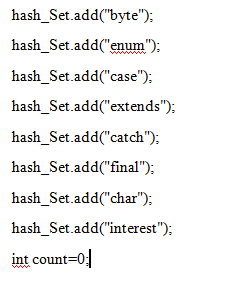
In this practical , we can learn Treeset , HahMap, Map , Map.put methods in brief.

**8.4**

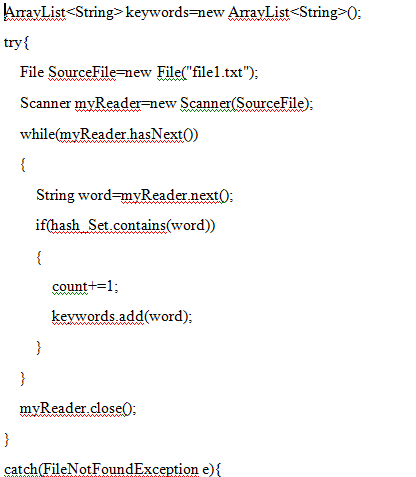
**AIM:** Write a code which counts the number of the keywords in a Java source file. Store all the keywords in a HashSet and use the contains () method to test if a word is in the keyword set.

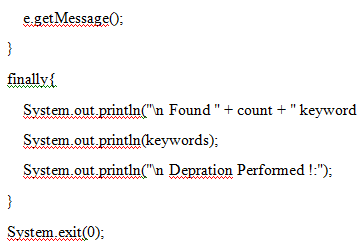
**Program:**

****

****

****

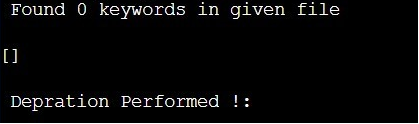
****

****

}

}

**Output :**

****

**Conclusion :**

In this practicale , we can learn about hash\_Set.add methods.