**JAVA Assignment 3: MODULE 2**

**Name – CHINMAYA GARNAIK**

**Class - FYMCA(B)**

**PRN – 1132220942**

1)Write a RMI application to do the following:                                        

Client will accept a string from user and communicate to the server.

Client will invoke find Vowel Count – remote method  
   on server, using accepted string and display vowel count from that string.

**VowelCountServer.java:-**

import java.rmi.registry.\*;

public class VowelCountServer {

public static void main(String[] args) {

try {

VowelCountImpl vc = new VowelCountImpl();

Registry registry = LocateRegistry.createRegistry(1550);

registry.bind("VowelCount", vc);

System.out.println("Server ready");

} catch (Exception e) {

System.out.println(e);

}

}

}

**VowelCount.java:-**

import java.rmi.\*;

public interface VowelCount extends Remote {

public int findVowelCount(String str) throws RemoteException;

}

**VowelCountClient.java:-**

import java.rmi.\*;

import java.util.Scanner;

public class VowelCountClient {

public static void main(String[] args) {

try {

String url = "rmi://localhost:1550/VowelCount";

VowelCount vc = (VowelCount) Naming.lookup(url);

Scanner sc = new Scanner(System.in);

System.out.println("Enter a string: ");

String str = sc.nextLine();

int count = vc.findVowelCount(str);

System.out.println("Number of vowels: " + count);

} catch (Exception e) {

System.out.println(e);

}

}

}

**VowelCountImpl.java:-**

import java.rmi.\*;

import java.rmi.server.\*;

public class VowelCountImpl extends UnicastRemoteObject implements VowelCount {

public VowelCountImpl() throws RemoteException {

super();

}

public int findVowelCount(String str) throws RemoteException {

int count = 0;

String strr = str.toLowerCase();

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

char ch = strr.charAt(i);

if (ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o' || ch == 'u') {

count++;

}

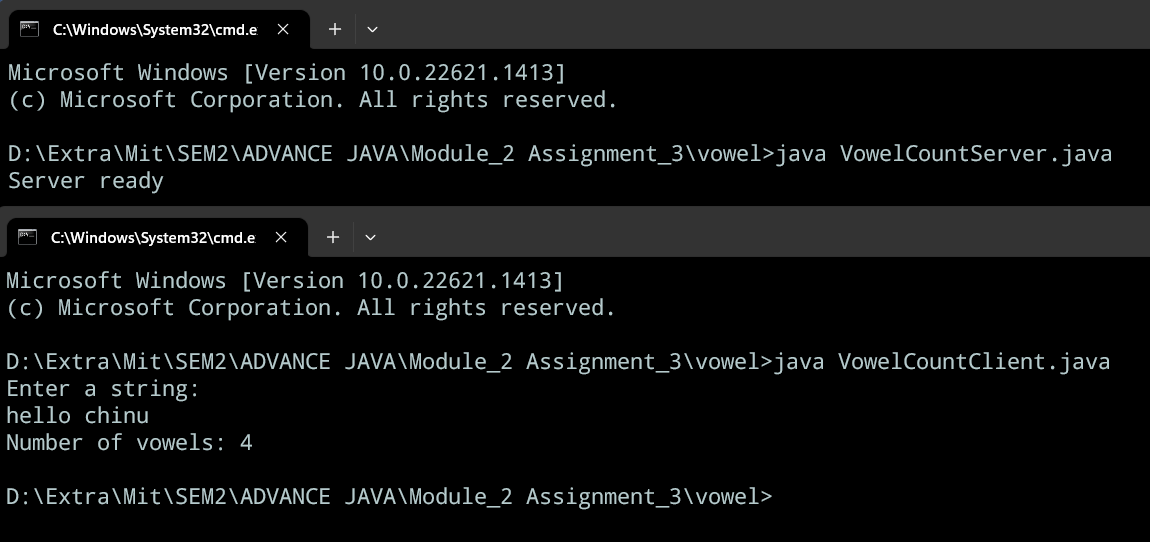
}

return count;

}

}

OUTPUT:-



2) Write all required classes or interfaces.

It is difficult to provide a comprehensive list of all classes or interfaces required in Java as it would depend on the specific needs of a project or application. However, some of the commonly used classes and interfaces in Java include:

* Object: the root class of all Java classes.
* String: used to represent text as a sequence of characters.
* List: an interface that defines a collection for storing elements in a specific order.
* Map: an interface that defines a collection for storing key-value pairs.
* Set: an interface that defines a collection for storing unique elements.
* Collection: an interface that defines the basic operations for working with collections.
* Iterator: an interface that provides a way to iterate over elements in a collection.
* Comparable: an interface that allows objects to be compared to each other.
* Runnable: an interface that defines a single method for running a task in a separate thread.
* Thread: a class that represents a separate thread of execution.
* InputStream: an abstract class for reading input streams of bytes.
* OutputStream: an abstract class for writing output streams of bytes.
* FileReader: a class for reading text files.
* FileWriter: a class for writing text files.
* Date: a class for working with dates and times.
* Calendar: a class for working with dates and times in a more flexible way than Date.
* Math: a class that provides mathematical functions.
* Random: a class that generates random numbers.
* Exception: a class that represents an error or exception in the program.
* Serializable: an interface that allows objects to be serialized and deserialized.

This is just a small selection of the many classes and interfaces available in Java. The specific requirements of a project or application will determine which classes and interfaces are needed.

3)  Write a Program to implement stack using RMI.

**Stck.java:-**

import java.rmi.\*;

public interface stck extends Remote {

public void push(int item) throws Exception;

public int pop() throws Exception;

public int peek() throws Exception;

public boolean isEmpty() throws RemoteException;

}

**Stckimpl.java:-**

import java.rmi.\*;

import java.rmi.server.\*;

import java.util.Stack;

public class stckimpl extends UnicastRemoteObject implements stck {

private Stack<Integer> stack;

public stckimpl() throws RemoteException {

super();

stack = new Stack<Integer>();

}

public void push(int item) throws RemoteException {

stack.push(item);

}

public int pop() throws RemoteException {

return stack.pop();

}

public int peek() throws RemoteException {

return stack.peek();

}

public boolean isEmpty() throws RemoteException {

return stack.isEmpty();

}

}

**Stckserver.java:-**

import java.rmi.registry.\*;

public class stckserver {

public static void main(String[] args) {

try {

stckimpl sti = new stckimpl();

Registry registry = LocateRegistry.createRegistry(1450);

registry.bind("stck", sti);

System.out.println("Server Ready");

} catch (Exception e) {

System.out.println(e);

}

}

}

**Stckclient.java:-**

import java.rmi.\*;

public class stckclient {

public static void main(String[] args) {

try {

String url = "rmi://localhost:1450/stck";

stck sti = (stck) Naming.lookup(url);

sti.push(1);

sti.push(2);

sti.push(3);

System.out.println(sti.pop()); // should print 3

System.out.println(sti.peek()); // should print 2

System.out.println(sti.isEmpty()); // should print false

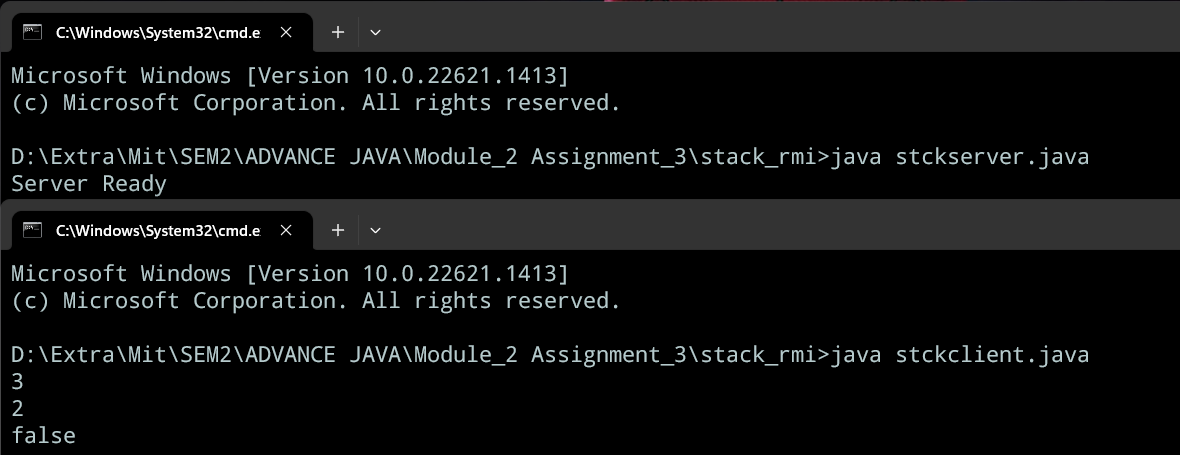
} catch (Exception e) {

System.out.println(e);

}

}

OUTPUT:-



4) Write a program to print sum of diagonal elements of array using RMI.

**Matrixmul.java:-**

import java.rmi.\*;

public interface matrixmul extends Remote {

public String multi(int a[][]) throws RemoteException;

public String showmatrix(int a[][]) throws RemoteException;

}

**Matrixmulimpl.java:-**

import java.rmi.\*;

import java.rmi.server.\*;

// import java.io.\*;

// import java.util.\*;

public class matrixmulimpl extends UnicastRemoteObject implements matrixmul {

public matrixmulimpl() throws RemoteException {

super();

}

public String multi(int a[][]) throws RemoteException {

int suml = 0, sumr = 0;

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

for (int j = 0; j < a.length; j++) {

if (i == j) {

suml += a[i][j];

}

if (i + j == a.length - 1) {

sumr += a[i][j];

}

}

}

return ("\nThe sum of the diagonal elements of the matrix you entered is: " + suml

+ "\n The sum of min diagonal elements is " + sumr);

}

public String showmatrix(int a[][]) throws RemoteException {

StringBuilder sb = new StringBuilder();

sb.append("The matrix is ..>>\n");

for (

int i = 0; i < (a.length); i++) {

sb.append("\n");

for (int j = 0; j < (a.length); j++) {

sb.append(a[i][j] + " ");

}

}

return sb.toString();

}

}

**Matrixmulserver.java:-**

import java.rmi.registry.\*;

public class matrixmulserver {

public static void main(String[] args) {

try {

matrixmulimpl mmi = new matrixmulimpl();

Registry registry = LocateRegistry.createRegistry(1500);

registry.bind("matrixmul", mmi);

System.out.println("Server Ready");

} catch (Exception e) {

System.out.println(e);

}

}

}

**Matrixmulclient.java:-**

import java.rmi.Naming;

import java.util.\*;

public class matrixmulclient {

public static void main(String[] args) {

try {

String url = "rmi://localhost:1500/matrixmul";

matrixmul mmi = (matrixmul) Naming.lookup(url);

Scanner sc = new Scanner(System.in);

System.out.println("Please enter the order of the matrix");

int n = sc.nextInt();

int a[][] = new int[n][n];

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

for (int j = 0; j < n; j++) {

System.out.println("Enter the element [" + i + "][" + j + "]");

a[i][j] = sc.nextInt();

}

}

System.out.println(mmi.showmatrix(a));

System.out.println(mmi.multi(a));

} catch (Exception e) {

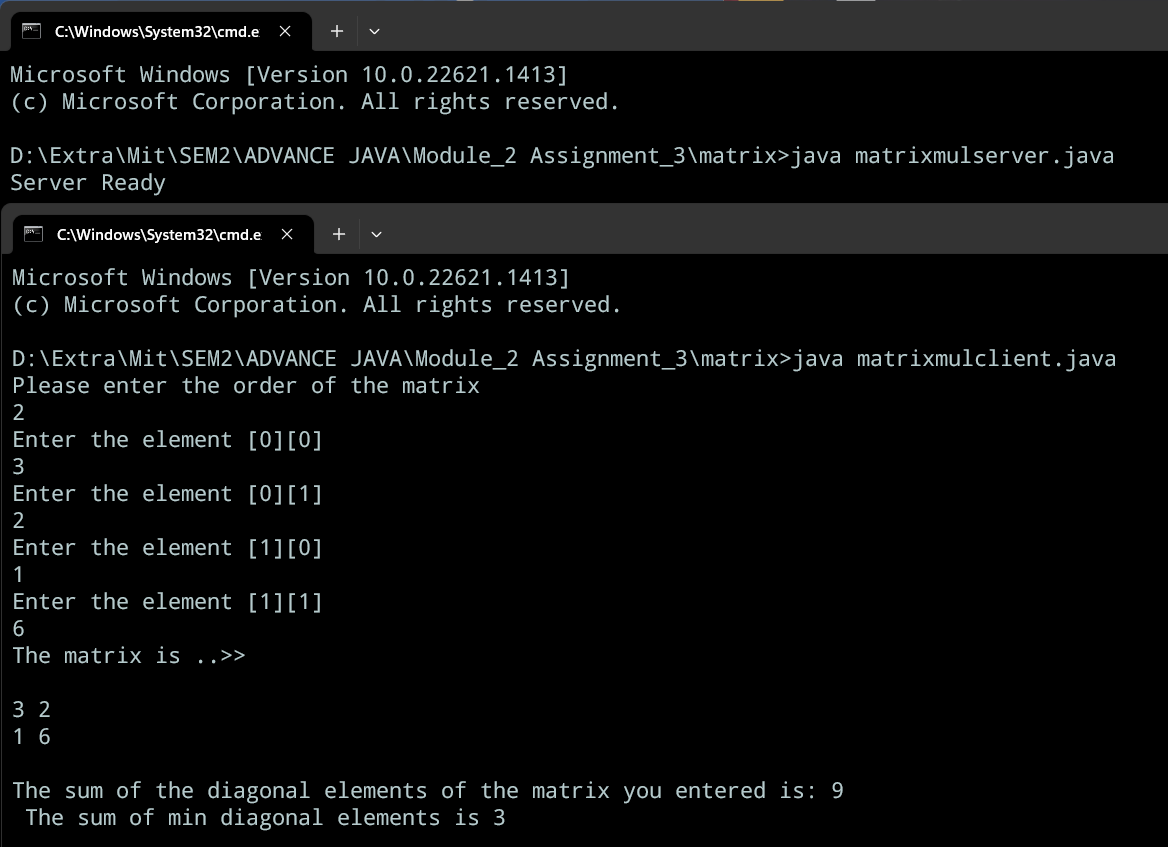
System.out.println(e);

}

}

}

OUTPUT:-



5) Write a Program to reverse string using RMI.

**Stringreverse.java:-**

import java.rmi.\*;

public interface stringreverse extends Remote {

public String ReverseString(String str) throws RemoteException;

}

**Stringreverseimpl.java:-**

import java.rmi.\*;

import java.rmi.server.\*;

public class stringreverseimpl extends UnicastRemoteObject implements stringreverse {

public stringreverseimpl() throws RemoteException {

super();

}

public String ReverseString(String str) throws RemoteException {

StringBuffer sbt = new StringBuffer(str);

sbt.reverse();

String str2 = sbt.toString();

return str2;

}

}

**Stringreverseserver.java:-**

import java.rmi.registry.\*;

public class stringreverseserver {

public static void main(String args[]) {

try {

stringreverseimpl sri = new stringreverseimpl();

Registry registry = LocateRegistry.createRegistry(1450);

registry.bind("stringreverse", sri);

System.out.println("server ready");

} catch (Exception e) {

System.out.println(e);

}

}

}

**Stringreverseclient.java:-**

import java.rmi.\*;

import java.util.Scanner;

public class stringreverseclient {

public static void main(String args[]) {

try {

String url = "rmi://localhost:1450/stringreverse";

stringreverse sri = (stringreverse) Naming.lookup(url);

Scanner sc = new Scanner(System.in);

System.out.println("Please enter the string you want to reverse");

String dmo = sc.nextLine();

String lmo = sri.ReverseString(dmo);

System.out.println("Reversed String is : " + lmo);

} catch (Exception e) {

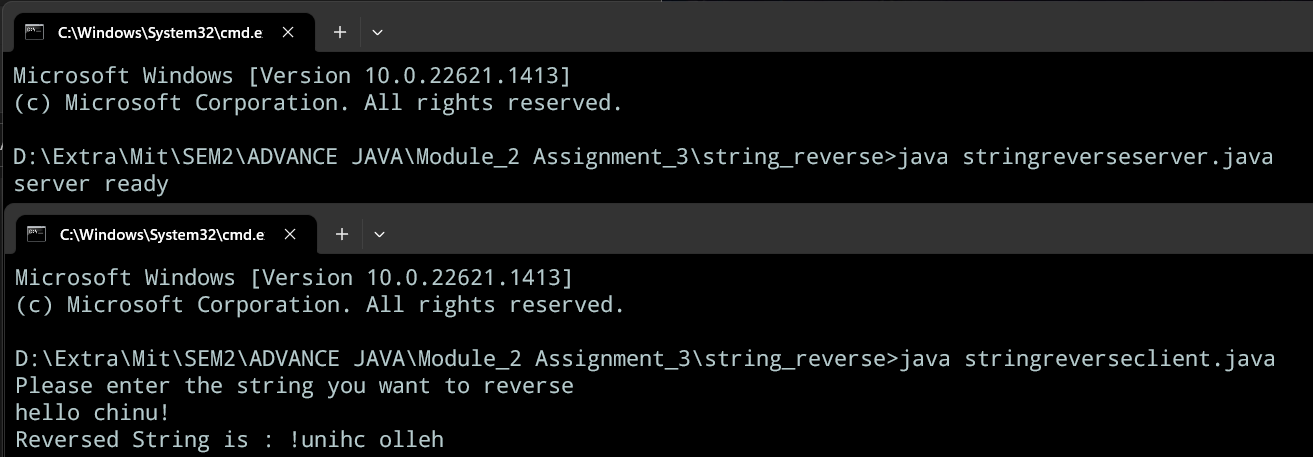
System.out.println(e);

}

}

}

OUTPUT:-



6) Write a program to read file and count no of words, characters and special symbols using RMI.

**Flcount.java:-**

import java.rmi.\*;

public interface flcount extends Remote {

public String counters(String str) throws RemoteException;

}

**Flcountimpl.java:-**

import java.rmi.\*;

import java.rmi.server.\*;

import java.io.\*;

public class flcountimpl extends UnicastRemoteObject implements flcount {

public flcountimpl() throws RemoteException {

super();

}

public String counters(String str) throws RemoteException {

File file = new File(str);

int c = 0, w = 0, s = 0;

String drs = "";

try {

BufferedReader br = new BufferedReader(new FileReader(file));

String st;

while ((st = br.readLine()) != null) {

String sr = st;

String words[] = sr.split(" ");

w = w + words.length;

for (String word : words)

c = c + word.length();

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

char ee = st.charAt(i);

if (!Character.isLetterOrDigit(ee) && !Character.isWhitespace(ee)) {

s++;

}

}

}

drs = "Number of words in the given file: " + w + "\nNumber of characters in the given file: " + c

+ " and the number of special characters in the file is " + s;

} catch (FileNotFoundException e) {

drs = "File not found";

} catch (IOException e) {

drs = "Error reading file";

} catch (Exception e) {

drs = "Unexpected error: " + e.getMessage();

}

return drs;

}

}

**Flcountserver.java:-**

import java.rmi.registry.\*;

public class flcountserver {

public static void main(String[] args) {

try {

flcountimpl fci = new flcountimpl();

Registry registry = LocateRegistry.createRegistry(1440);

registry.bind("flcount", fci);

System.out.println("Server Ready");

} catch (Exception e) {

System.out.println(e);

}

}

}

**Flcountclient.java:-**

import java.rmi.\*;

import java.util.Scanner;

public class flcountclient {

public static void main(String[] args) {

try {

String url = "rmi://localhost:1440/flcount";

flcount fci = (flcount) Naming.lookup(url);

Scanner sc = new Scanner(System.in);

System.out.println("Please enter the file name you want to count the characters of");

String fname = sc.nextLine();

System.out.println(fci.counters(fname));

} catch (Exception e) {

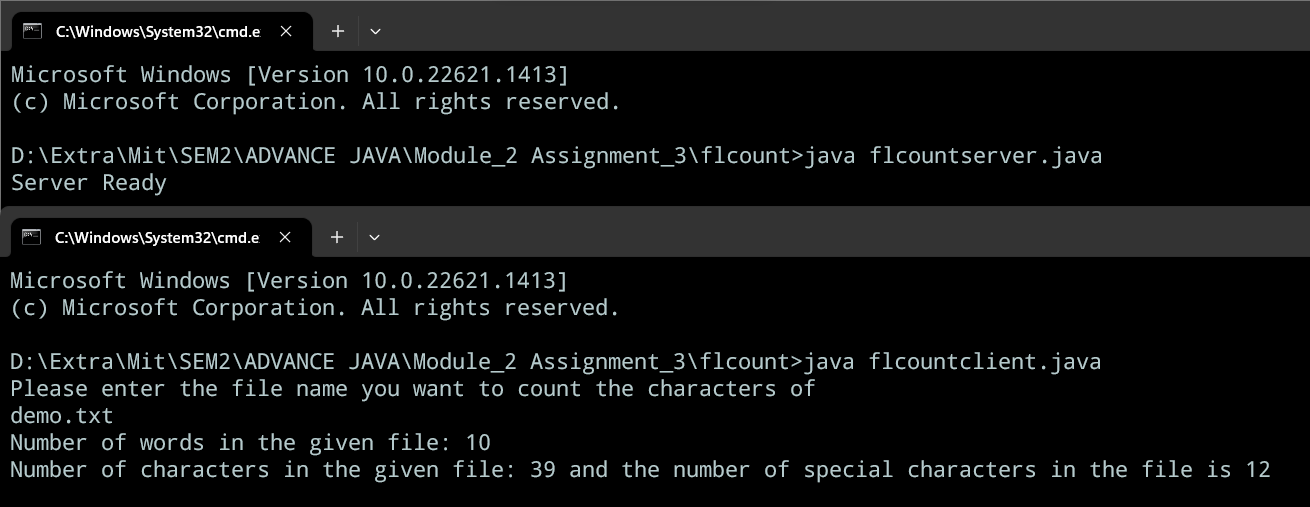
System.out.println(e);

}

}

}

OUTPUT:-

****