**Department of Computer Science & Engineering, SDMCET, Dharwad-2**

****

**AOOP Assignment 1 Submission Report**

**[Submitted as part of CTA Assignment No-1]**

**Course: Advanced Object-Oriented Programming Course**

**Code: 18UCSE508**

**Semester: V**

**Division:A**

**Submitted by: Aachan B kulkarni**

**USN:2SD20CS002**

1. **Problem Definition:**

**Write a Java program to generate and handle any three built-in exceptions and display appropriate**

**error messages.**

1. **Problem Definition:**

**Write a Java program to read an integer and check whether the number is prime or not. If negative**

**number is entered, throw an exception NegativeNumberNotAllowedException and if entered**

**number is not prime, then throw NumberNotPrimeException.**

1. **Problem Definition:**

**Write a Java program to perform the following operations:**

**a) Read a line of text**

**b) Search for a sub-string SDMCET (case insensitive search)**

**c) If found, then print success message**

**d) Otherwise throw an exception SubStringNotFoundException with appropriate message.**

1. **Problem Definition:**

**Write a Java program to perform the following operations:**

**a) Create a file named Alphabets.txt and insert appropriate data into it**

**b) Read the file and copy all the consonants into another file named Consonants.txt**

**c) If vowel is encountered, throw an exception VowelNotAllowedException and continue until**

**end of file.**

1. **Problem Definition:** **Write a Java program to implement the following scenario:**
2. **Create a file named Integers.txt and insert n-random integers into it**

**b) Create three threads T1, T2 and T3 that read n/3 integers in sequence of occurrence of**

**numbers from the file and sort the read n/3 integers.**

1. **Thread T4 waits for all the threads T1, T2 and T3 to complete sorting, then sorts and outputs**

**the entire list of sorted numbers to another file named SortedIntegers.txt**

**1.Problem Definition:**

**Write a Java program to generate and handle any three built-in exceptions and display appropriate error messages.**

**Code part:-**

**import java.io.\*;**

**class Tm1{**

**public static void main(String args[]){**

**try{ //first exception occured**

**int a=30,b=0;**

**int c=a/b;**

**System.out.println("Result=" + c);**

**}catch(ArithmeticException e){**

**System.out.println("Can't divide a number by 0");**

**}**

**try{ //second exception occured**

**int z[] = new int[4];**

**z[7]=9;**

**}catch(ArrayIndexOutOfBoundsException e){**

**System.out.println("Array Index is Out Of bounds");**

**}**

**try{ //third exception occured**

**FileInputStream f = null;**

**f = new FileInputStream("abc.txt");**

**int i;**

**while(( i =f.read())!=-1){**

**System.out.println((char)i);**

**}**

**f.close();**

**}catch(IOException e){**

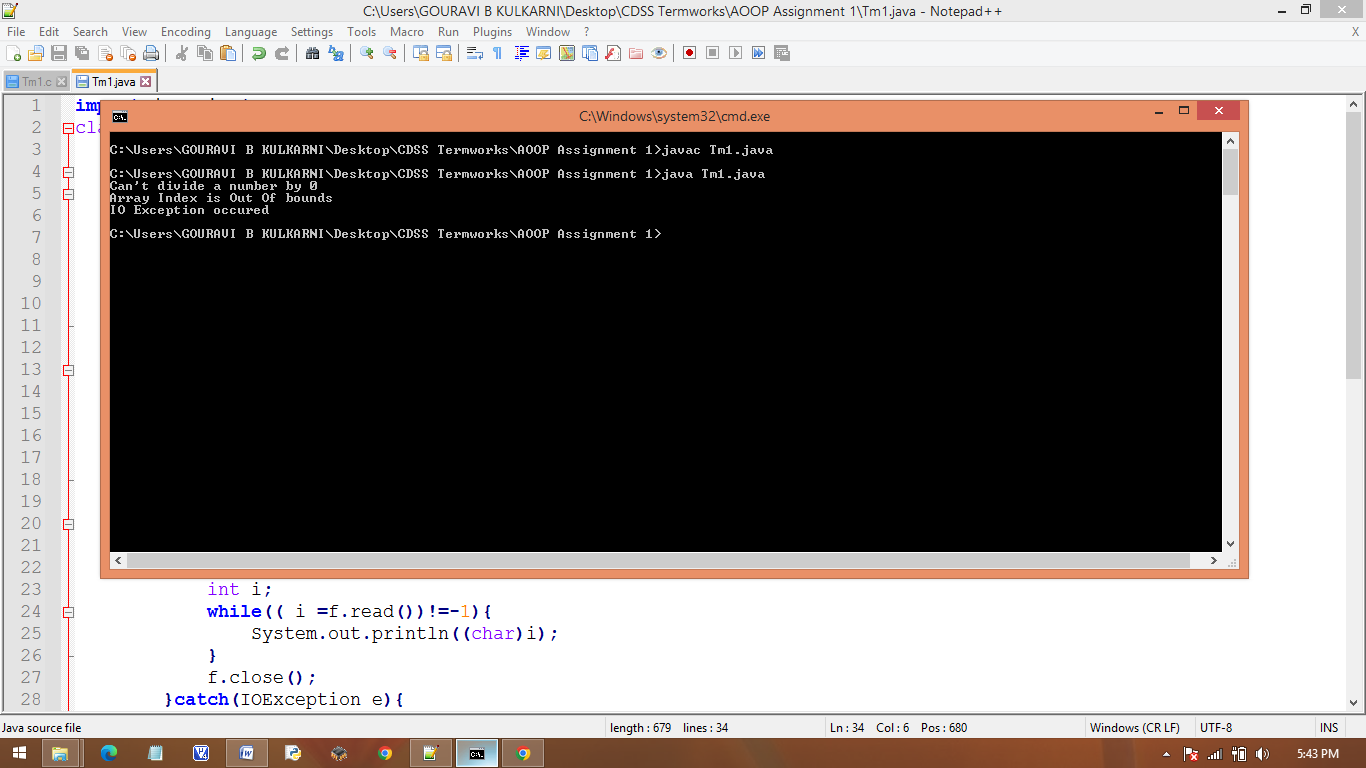
**System.out.println("IO Exception occured");**

**}**

**}**

**}**

**Screen Shots of Execution:**

****

**2.Problem Definition:**

**Write a Java program to read an integer and check whether the number is prime or not. If negative**

**number is entered, throw an exception NegativeNumberNotAllowedException and if entered**

**number is not prime, then throw NumberNotPrimeException.**

**Code part:-**

**import java.util.Scanner;**

**public class Tm2{**

**public static void main(String args[]){**

**Scanner sc = new Scanner(System.in);**

**System.out.println("Enter a number");**

**int num = sc.nextInt();**

**System.out.println("Entered number is " + num);**

**System.out.println("............................");**

**try{**

**if(num<0){**

**throw new NegativeNumberNotAllowedException("Negative number is entered");**

**}**

**if(num>=0){**

**for(int i=2;i<num/2;i++)**

**{**

**if((num%i)==0)**

**throw new NumberNotPrimeException("Number is not a prime");**

**break;**

**}**

**}**

**System.out.println("Entered number is prime" + num);**

**}catch( NegativeNumberNotAllowedException np){ //catch handles the exception**

**System.out.println( np.toString());**

**}catch( NumberNotPrimeException npn){ //catch handles the exception**

**System.out.println( npn.toString());**

**}**

**}**

**}**

**class NegativeNumberNotAllowedException extends Exception{**

**public NegativeNumberNotAllowedException(String message){**

**super(message);**

**}**

**}**

**class NumberNotPrimeException extends Exception{**

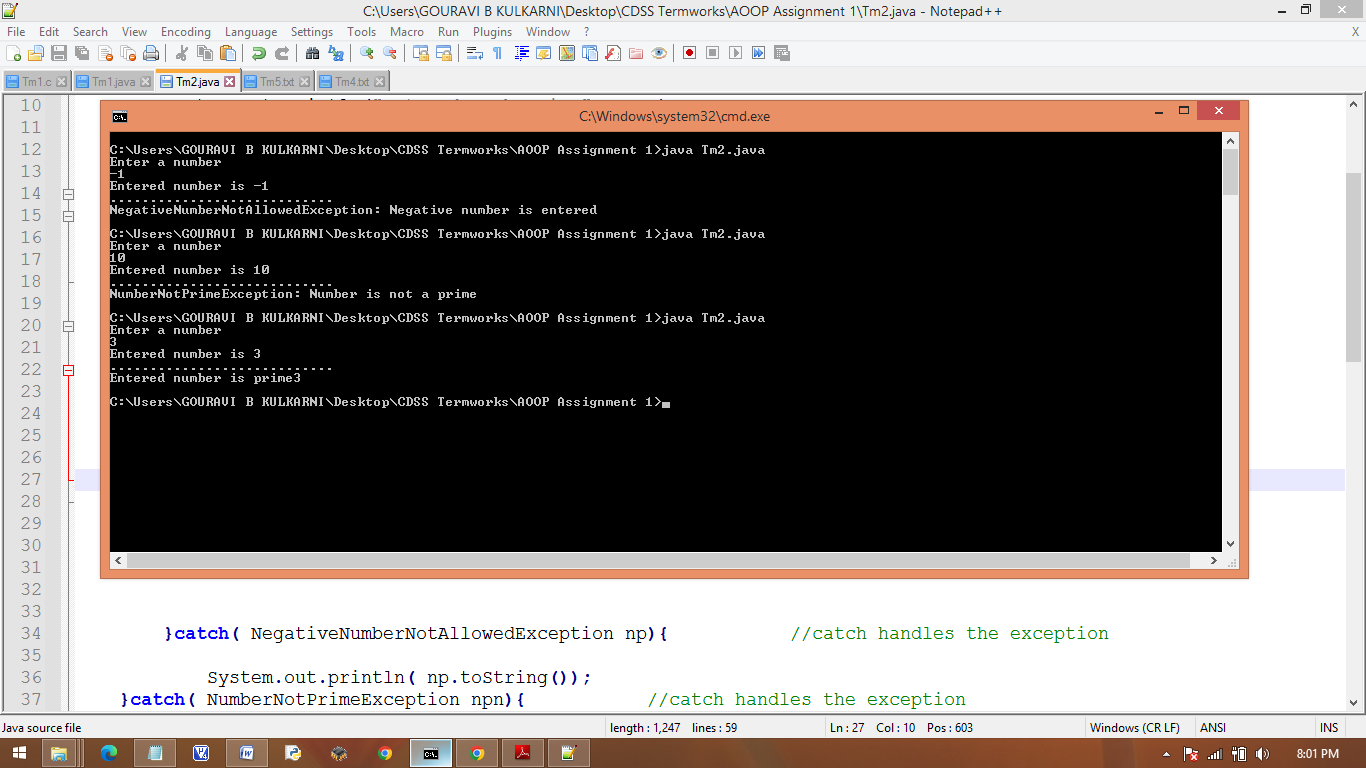
**public NumberNotPrimeException(String message){**

**super(message);**

**}**

**}**

**Screen Shots of Execution:**

****

**3.Problem Definition:**

**Write a Java program to perform the following operations:**

**a) Read a line of text**

**b) Search for a sub-string SDMCET (case insensitive search)**

**c) If found, then print success message**

**d) Otherwise throw an exception SubStringNotFoundException with appropriate message.**

**Code part:-**

**import java.util.Scanner;**

**public class Tm3{**

**public static void main(String args[]){**

**Scanner sc = new Scanner(System.in);**

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

**String str = sc.nextLine();**

**System.out.println("Entered string is " + str);**

**System.out.println("............................");**

**try{**

**if(str.contains("SDMCET")){**

**System.out.println("Search for sub String SDMCET sucessful");**

**}**

**else{**

**throw new SubStringNotFoundException("Sub string not found");**

**}**

**}catch( SubStringNotFoundException np){ //catch handles the exception**

**System.out.println( np.toString());**

**}**

**}**

**}**

**class SubStringNotFoundException extends Exception{**

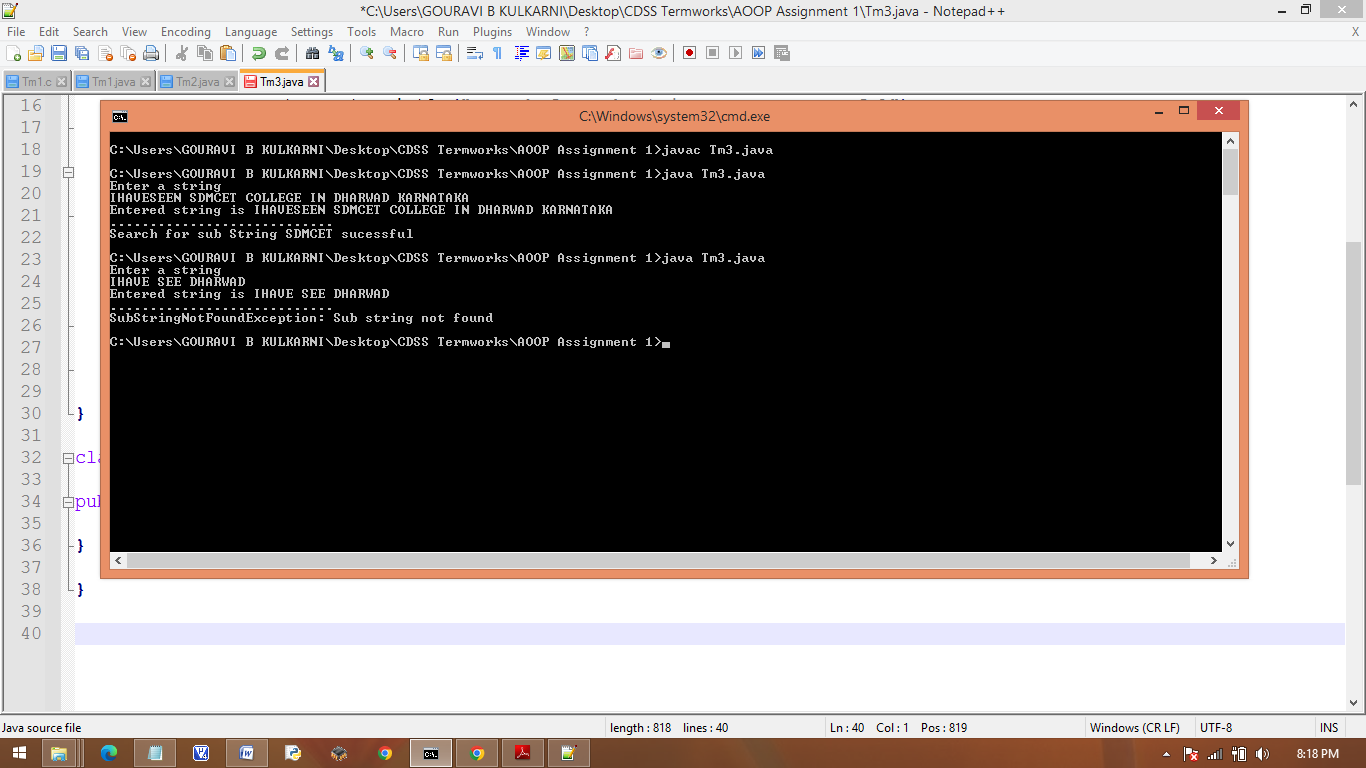
**public SubStringNotFoundException(String message){**

**super(message);**

**}**

**}**

**Screen Shots of Execution:**

****

1. **Problem Definition:**

**Write a Java program to perform the following operations:**

**a) Create a file named Alphabets.txt and insert appropriate data into it**

**b) Read the file and copy all the consonants into another file named Consonants.txt**

**c) If vowel is encountered, throw an exception VowelNotAllowedException and continue until**

**end of file.**

**Code part:-**

**//copy all consonants from file to another, if vowel is encountered throw exception**

**import java.util.Scanner;**

**import java.io.\*;**

**//main class**

**public class Tm4 {**

**static Scanner in = new Scanner(System.in);**

**//main method**

**public static void main(String args[]) throws IOException {**

**try (BufferedReader buf = new BufferedReader(new FileReader("Alphabets.txt"))) {**

**try (BufferedWriter wrt = new BufferedWriter(new FileWriter("Consonants.txt"))) {**

**String str = buf.readLine();**

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

**try {**

**if(str.charAt(i)=='a'||str.charAt(i)=='e'||str.charAt(i)=='i'||str.charAt(i)=='o'||str.charAt(i)=='u')**

**throw new VowelNotAllowedException("Not a consonant");**

**else {**

**wrt.write(str.charAt(i));**

**}**

**}catch(VowelNotAllowedException e) {**

**System.out.println("File is added successfully with the content.");**

**System.out.println("Characters not added "+str.charAt(i));**

**}**

**}**

**}**

**}**

**}//End of main method**

**}//End of main class**

**//VowelNotAllowedException class**

**class VowelNotAllowedException extends Exception{**

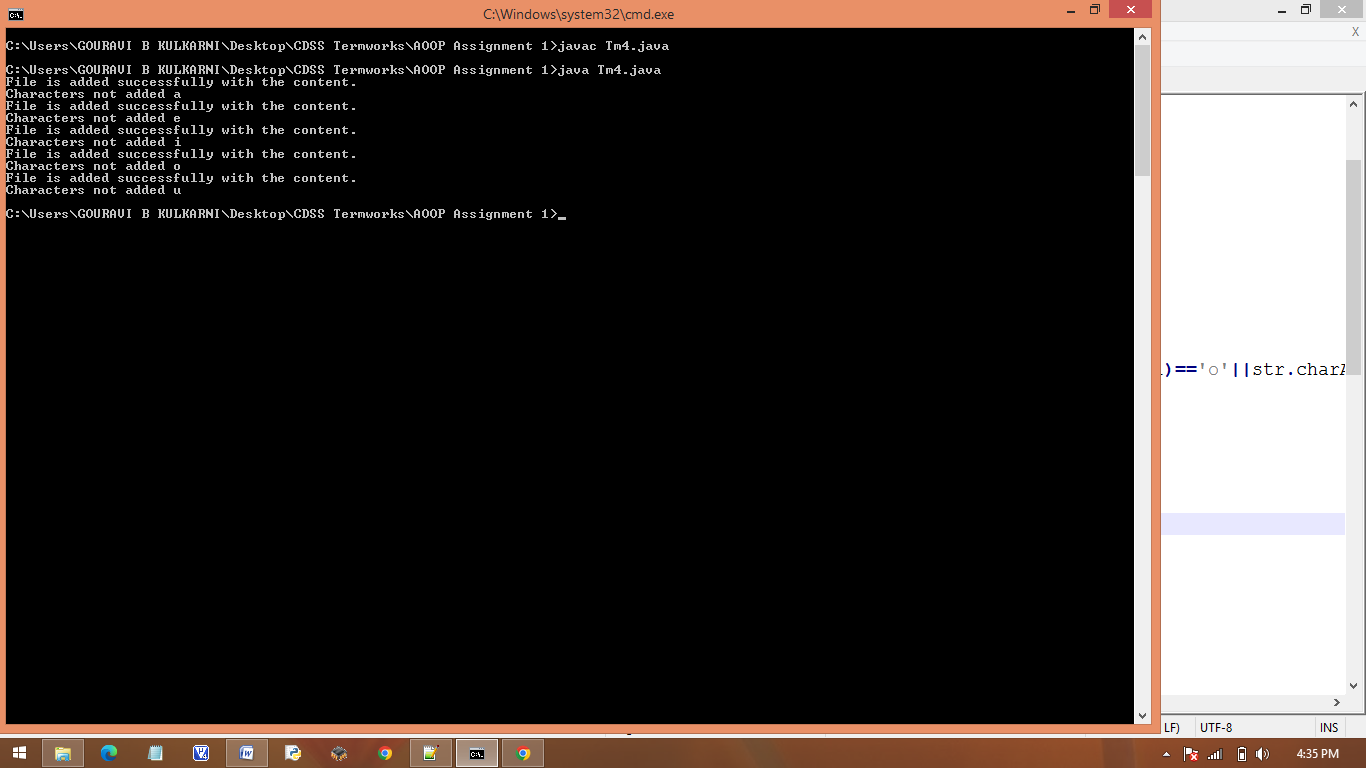
**public VowelNotAllowedException(String message) {**

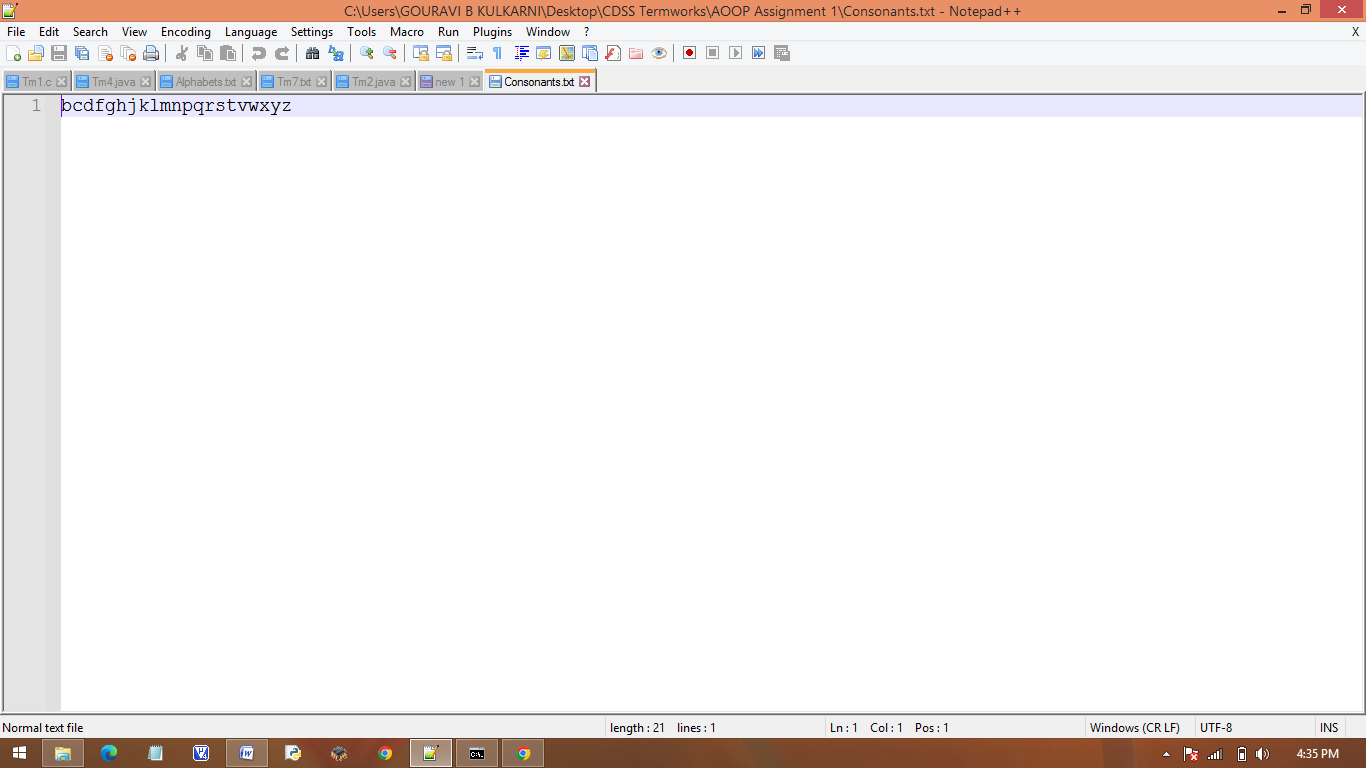
**super(message);**

**}**

**}**

**Screen Shots of Execution:**

****

****

Consonants.txt

1. **Problem Definition:** **Write a Java program to implement the following scenario:**
2. **Create a file named Integers.txt and insert n-random integers into it**

**b) Create three threads T1, T2 and T3 that read n/3 integers in sequence of occurrence of**

**numbers from the file and sort the read n/3 integers.**

1. **Thread T4 waits for all the threads T1, T2 and T3 to complete sorting, then sorts and outputs**

**the entire list of sorted numbers to another file named SortedIntegers.txt**

**Code part:-**

**//Program to Sort 'n' numbers with 3 threads and combine to give the final sorted list**

**import java.io.\*;**

**import java.util.\*;**

**//main class**

**public class Tm5 {**

**public static ArrayList<Integer> alist1 = new ArrayList<Integer>();**

**public static ArrayList<Integer> alist2 = new ArrayList<Integer>();**

**public static ArrayList<Integer> alist3 = new ArrayList<Integer>();**

**public static ArrayList<Integer> sortedlist = new ArrayList<Integer>();**

**static Scanner in = new Scanner(System.in);**

**//main method**

**public static void main(String args[]) throws IOException{**

**try(BufferedReader buf = new BufferedReader(new FileReader("Integers.txt"))){**

**try(BufferedWriter wrt = new BufferedWriter(new FileWriter("SortedIntegers.txt"))){**

**String str = buf.readLine();**

**String word[] = str.split(" ");**

**ArrayList<Integer> arlist = new ArrayList<Integer>();**

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

**arlist.add(Integer.parseInt(word[i]));**

**}**

**//splitting into 3 sub arrays to be fed into threads**

**ArrayList<Integer> arlist1 = new ArrayList<Integer>();**

**ArrayList<Integer> arlist2 = new ArrayList<Integer>();**

**ArrayList<Integer> arlist3 = new ArrayList<Integer>();**

**for(int i=0; i<(arlist.size()/3);i++) {**

**arlist1.add(arlist.get(i));**

**}**

**for(int i=(arlist.size()/3);i<(2\*(arlist.size()/3));i++) {**

**arlist2.add(arlist.get(i));**

**}**

**for(int i=(2\*(arlist.size()/3));i<(arlist.size());i++) {**

**arlist3.add(arlist.get(i));**

**}**

**//Creating threads**

**Thread\_1 T1 = new Thread\_1(arlist1);**

**Thread\_2 T2 = new Thread\_2(arlist2);**

**Thread\_3 T3 = new Thread\_3(arlist3);**

**try {**

**T1.start();**

**T1.join();**

**T2.start();**

**T2.join();**

**T3.start();**

**T3.join();**

**}catch(Exception e) {**

**System.out.println("Error: " + e.toString());**

**}**

**arlist1.addAll(arlist2);**

**arlist1.addAll(arlist3);**

**Thread\_4 T4 = new Thread\_4(arlist1);**

**try {**

**T4.start();**

**T4.join();**

**}catch(Exception e) {**

**System.out.println("Error: " + e.toString());**

**}**

**}**

**}**

**}**

**public static ArrayList<Integer> sortlist(ArrayList<Integer> arraylist) {**

**Collections.sort(arraylist);**

**return arraylist;**

**}**

**}**

**class Thread\_1 extends Thread {**

**public ArrayList<Integer> alist11;**

**public ArrayList<Integer> alist21;**

**Thread\_1(){**

**}**

**Thread\_1(ArrayList<Integer> alist11){**

**this.alist11 = new ArrayList<>(List.copyOf(alist11));**

**}**

**public void run() {**

**alist21 = Tm5.sortlist(alist11);**

**}**

**}**

**class Thread\_2 extends Thread {**

**public ArrayList<Integer> alist12;**

**public ArrayList<Integer> alist22;**

**Thread\_2(){**

**}**

**Thread\_2(ArrayList<Integer> alist12){**

**this.alist12 = new ArrayList<>(List.copyOf(alist12));**

**}**

**public void run() {**

**alist22 = Tm5.sortlist(alist12);**

**}**

**}**

**class Thread\_3 extends Thread {**

**public ArrayList<Integer> alist13;**

**public ArrayList<Integer> alist23;**

**Thread\_3(){**

**}**

**Thread\_3(ArrayList<Integer> alist13){**

**this.alist13 = new ArrayList<>(List.copyOf(alist13));**

**}**

**public void run() {**

**alist23 = Tm5.sortlist(alist13);**

**}**

**}**

**class Thread\_4 extends Thread {**

**public ArrayList<Integer> alist14;**

**public ArrayList<Integer> alist24;**

**Thread\_4(){**

**}**

**Thread\_4(ArrayList<Integer> alist13){**

**this.alist14 = new ArrayList<>(List.copyOf(alist14));**

**}**

**public void run() {**

**alist24 = Tm5.sortlist(alist14);**

**try {**

**try (BufferedWriter wt = new BufferedWriter(new FileWriter("SortedIntegers.txt"))) {**

**for(int i=0; i<alist24.size();i++) {**

**wt.write(alist24.indexOf(i) + " ");**

**}**

**}**

**} catch (IOException e) {**

**e.printStackTrace();**

**}**

**}**

**}**

**Screen Shots of Execution:**

