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

AOOP Assignment Submission Report

[Submitted as part of CTA Assignment No-1]

Course:	Advanced Object-Oriented Programming	Course Code:	18UCSE508
Semester:	V	Division:	A

Submitted by:

USN:	2SD20CS012	Name:	AMAN A SHETTY

1. Problem Definition:

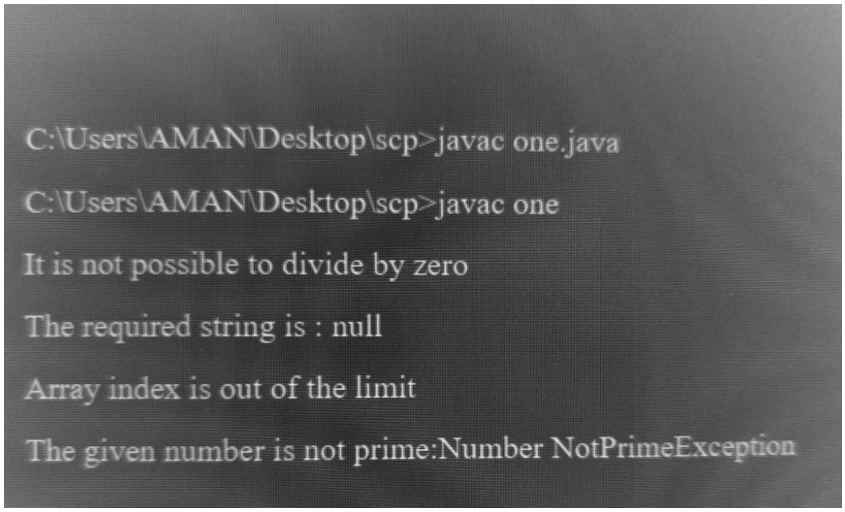
Q1. Write a Java program to generate and handle any three built-in exceptions and display appropriate error messages

2. Java Program:

```
//Program to demonstrate some of the built in exception handlers in java

//main class
public class Program {

	//main method
	public static void main(String args[]) {

		//Arithmetic Exception
		try {
			int a = 10, b=0;//Dividing by zero raises an exception
			int c = a/b;
			System.out.println("The result is: " + c);
		}catch(ArithmeticException e) {
			System.out.println("Cannot divide by zero");
		}

		//Null pointer Exception
		try {
			String str = null;//String is initialized to null
			System.out.println(" Required string is: " + str);
```

```
            }catch(NullPointerException e) {
                System.out.println(" String does not exist");
            }

            //Array index out of bounds Exception
            try {
                int arr[] = new int[5];
                arr[6] = 1;//insertion of an element out of the scope of the array
            }catch(ArrayIndexOutOfBoundsException  e) {
                System.out.println("Array index is out of the limit of the declared array");
            }
        }//End of main method
}//End of main class
```

3. Screen Shots of Execution:



1. Problem Definition:

Q2. 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.

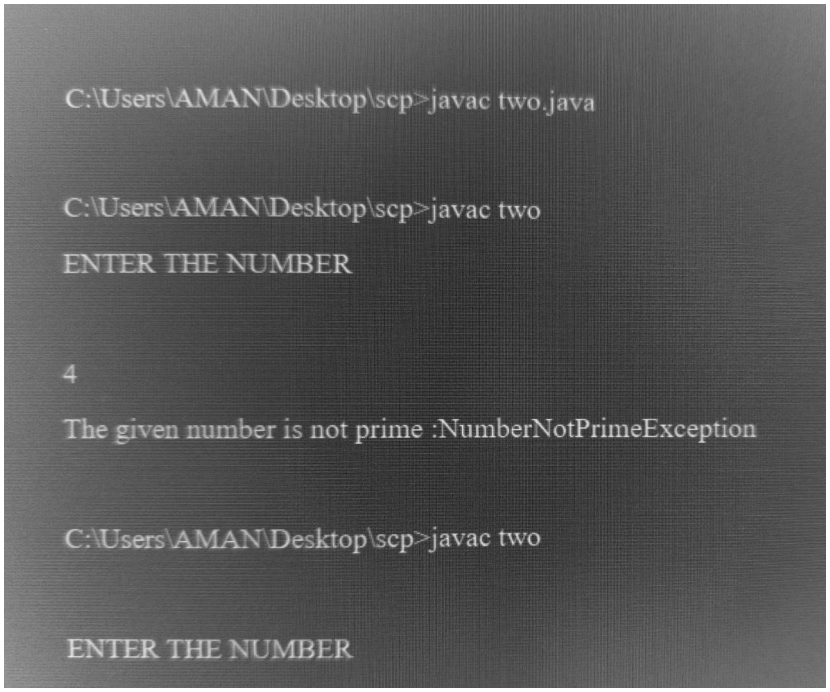
2. Java Program:

```
import java.util.*;

class NumberNotPrimeException extends Exception {

}

class NegativeNumberNotAllowedException extends Exception {
}

class two {

        public static void main(String args[]) {

                Scanner in=new Scanner(System.in);

                int n,flag=0;
                System.out.println("Enter Number\n");
```

```
            n=in.nextInt();



            try {

                if(n<0)
                {
                    throw new NegativeNumberNotAllowedException();
                }

            } catch(NegativeNumberNotAllowedException e) {
        System.out.println("Given number is negative :" +e.toString());
    }

            try {



                    if(n==0 || n==1)
                {
                        flag=1;
                }

                for(int i=2;i<=n/2;i++)
                {
                    if(n%i==0)
                    {
                        flag=1;
                        break;
                    }
```

```
                    }

                    if(flag==1)
                    {
                        throw new NumberNotPrimeException();
                    }


            } catch(NumberNotPrimeException p) {
        System.out.println(" Given number is not prime :" +p.toString());
    }
  }

}
```

3. Screen Shots of Execution:



1. Problem Definition:

Q3. 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

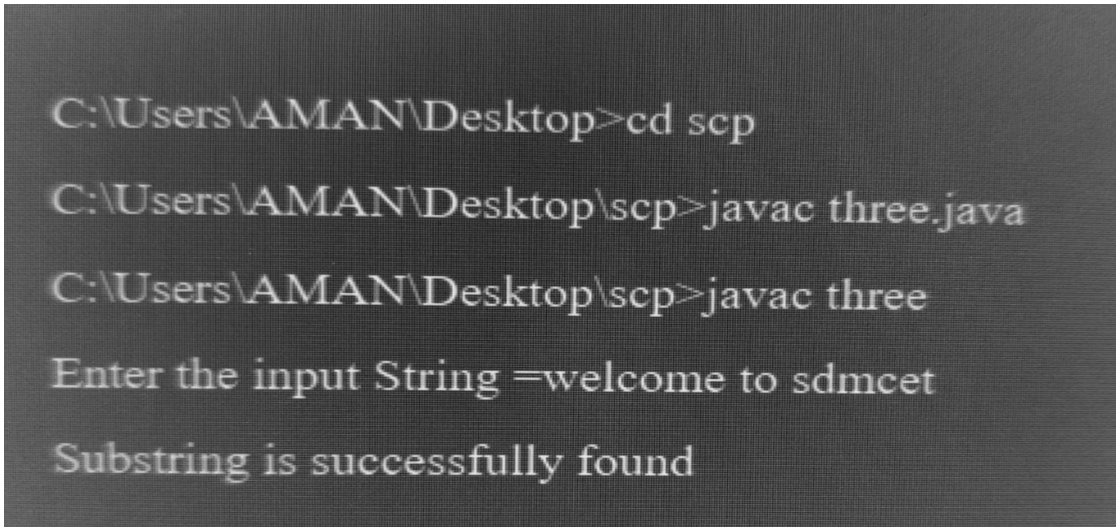
2. Java Program:

```
import java.util.*;
import java.util.Scanner;

/**
 * Assignment_3
 */
public class Assignment_3 {

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

    Scanner sc= new Scanner(System.in);
    System.out.print("Enter the String =");
    String testString = sc.nextLine();
  testString =  testString.toUpperCase();
    String subString="SDMCET";

    int i=0,j=0;
    while(i<testString.length()){
```

```
        if(testString.charAt(i)==subString.charAt(j) && j<subString.length()-1){
            i++;
            j++;
        }else{
            i++;
        }



    }  //end of while


    if(j == subString.length()-1){
        System.out.println("Substring is present");
    }else{
        throw new SubstringNotFoundException("Substring is not found !! please enter the valid
input");
    }



   }


}


class SubstringNotFoundException extends Exception{


    String str;
    SubstringNotFoundException(String str){
        this.str = str;
    }
    public String toString() {
```

```
        return this.str;
    }
}
```

3. Screen Shots of Execution:



1. Problem Definition:

Q4. 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

2. Java Program:

```
import java.util.Scanner;
import java.io.*;


public class Assignment Alphabets {

    public static void main(String[] args) {

        try{

            FileWriter w = new FileWriter("Alphabets.txt");
            Scanner sc = new Scanner(System.in);
            System.out.print("Enter the data to write in the file :");
            String str = sc.nextLine();
            w.write(str);
            w.close();
            File file = new File("Alphabets.txt");

            Scanner reader = new Scanner(file);
```

```
        StringBuilder s = new StringBuilder();
        FileWriter write = new FileWriter("Consonate.txt");
        while(reader.hasNext()){
         String data  = reader.next();
         for (int i = 0; i < data.length(); i++) {
           if(isVowel(data.charAt(i))){
               System.out.println("vowel found " + data.charAt(i));

           }else{

            s.append(data.charAt(i));
           }
         }
         write.write(s.toString());
        }



        write.close();

      }
    catch(VowelNotFoundException v){
      System.out.println("vowel found");
    }catch(FileNotFoundException e){
        System.out.println(e);

    }catch(IOException ex){
      System.out.println(ex);
    }
  }
```

```
    static boolean isVowel(char c) throws VowelNotFoundException{
        if(c == 'a' || c =='e' || c =='i' || c =='o' || c =='u' || c =='A' || c =='E' || c =='I'|| c =='O'|| c =='U'){
            return true;
        }else{
            return false;
        }
    }
    class VowelNotFoundException extends Exception{

    String str;
    VowelNotFoundException(String str){
        this.str = str;
    }
    public String toString() {
        return this.str;
    }

}

}
```

3. Screen Shots of Execution:

1. Problem Definition:

Q5. Write a Java program to implement the following scenario:

a) 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

c) 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

2. Java Program:

```
*/
import java.nio.*;
import java.util.Arrays;
import java.util.Scanner;
import java.io.*;
import java.lang.reflect.Array;
import java.net.Socket;

class MyThread extends Thread {

   MyThread(String name) {

      super(name);
   }

   public void readAndSort(Integer intArr[], int startIndex, int endIndex, int size, FileWriter
sortedIntWriter) {
```

```
        int[] threadArr = new int[size / 3];
        int i = 0;
        int j = startIndex;
        for (i = 0, j = startIndex; j < endIndex; i++, j++) {

            threadArr[i] = intArr[j];
        }

        Arrays.sort(threadArr);
        System.out.println("Sorted: " + Arrays.toString(threadArr));

        if (super.getName() == "t4") {

            try {

                sortedIntWriter.write(Arrays.toString(threadArr));
                sortedIntWriter.flush();
            } catch (IOException e) {

                e.printStackTrace();
            }
        }
    }
}

public class Q5 {

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

```
Scanner sc = new Scanner(System.in);
File intFile = new File("Integers");
File sortedInt = new File("SortedIntegers");
System.out.println("Enter total number of integers");
int n = sc.nextInt();

try {

    intFile.createNewFile();
    sortedInt.createNewFile();

} catch (IOException e) {

    System.out.println("IOException while creating file");
}

System.out.print("Enter " + n + " integers:");
int i = 0;
Integer[] intArr = new Integer[n];
try {

    FileWriter intWriter = new FileWriter(intFile);
    for (i = 0; i < n; i++) {

        intArr[i] = sc.nextInt();
        intWriter.write(Integer.toString(intArr[i]) + "\n");
        intWriter.flush();
    }
```

```
        System.out.println(Arrays.toString(intArr));

    } catch (IOException e) {

        e.printStackTrace();
    }

    MyThread t1 = new MyThread("t1");
    try {

        FileWriter sortedIntWriter = new FileWriter(sortedInt);
        t1.start();
        t1.readAndSort(intArr, 0, n / 3, n, sortedIntWriter);
        t1.join();
    } catch (InterruptedException e) {

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

    MyThread t2 = new MyThread("t2");

    try {

        FileWriter sortedIntWriter = new FileWriter(sortedInt);
        t2.start();
        t2.readAndSort(intArr, n / 3, 2 * n / 3, n, sortedIntWriter);
        t2.join();
```

```
} catch (InterruptedException e) {

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

MyThread t3 = new MyThread("t3");

try {

    FileWriter sortedIntWriter = new FileWriter(sortedInt);
    t3.start();
    t3.readAndSort(intArr, 2 * n / 3, n, n, sortedIntWriter);
    t3.join();

} catch (InterruptedException e) {

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

MyThread t4 = new MyThread("t4");
try {

    FileWriter sortedIntWriter = new FileWriter(sortedInt);
    t4.start();
    t4.readAndSort(intArr, 0, n, 3 * n, sortedIntWriter);
```

```
            t4.join();

        } catch (InterruptedException e) {

            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

3. Screen Shots of Execution: