



AOOP Assignment Submission Report

[Submitted as part of CTA Assignment No-1]

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

Submitted by:

USN:	2SD20CS011	Name:	Alok Hiremath
------	------------	-------	---------------

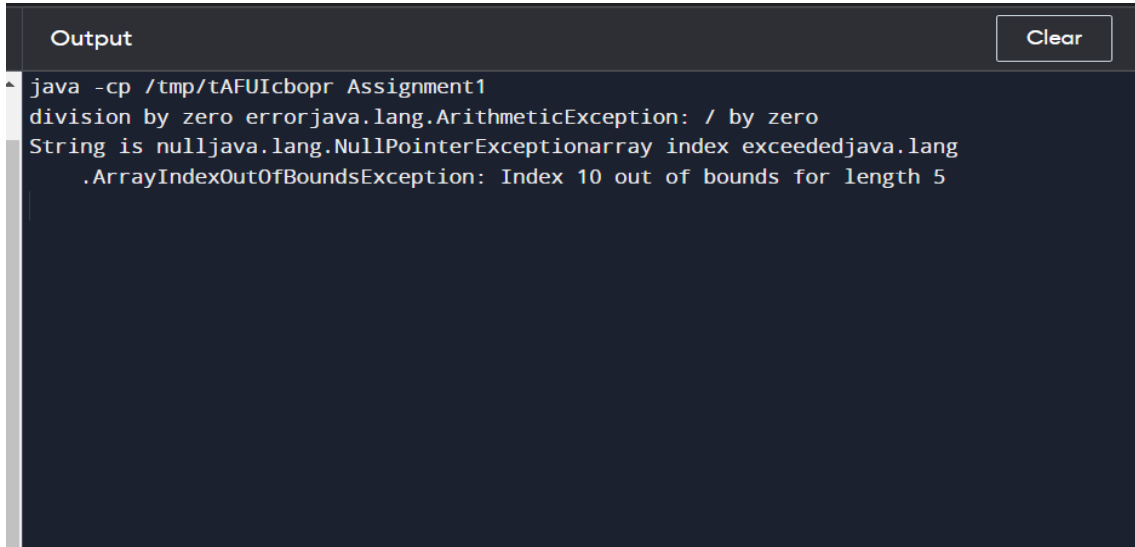
1. Problem definition:

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

2. Java Program:

```
public class Assignment1{  
    public static void main(String[] args){  
        int a=10;  
        int b=5;  
        int c=5;  
        String s=null;  
        int d[]=new int[5];  
        try{  
            System.out.println(a/(b-c));  
        } catch(ArithmeticException ae){  
            System.out.println("division by zero error"+ae);  
        }  
        try{  
            System.out.println(s.length());  
        }  
        catch(NullPointerException ne){  
            System.out.println("String is null"+ne);  
        }  
        try{  
            d[10]=50;  
        }  
        catch(ArrayIndexOutOfBoundsException aoe){  
            System.out.println("array index exceeded"+aoe);  
        }  
    }  
}
```

3. Screenshot of execution:



The screenshot shows a dark-themed IDE output window. The title bar reads 'Output' and has a 'Clear' button on the right. The output text is as follows:

```
java -cp /tmp/tAFUIcbopr Assignment1
division by zero errorjava.lang.ArithmeticException: / by zero
String is nulljava.lang.NullPointerExceptionarray index exceededjava.lang
.ArrayIndexOutOfBoundsException: Index 10 out of bounds for length 5
```

1.Problem definition:

2. 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:

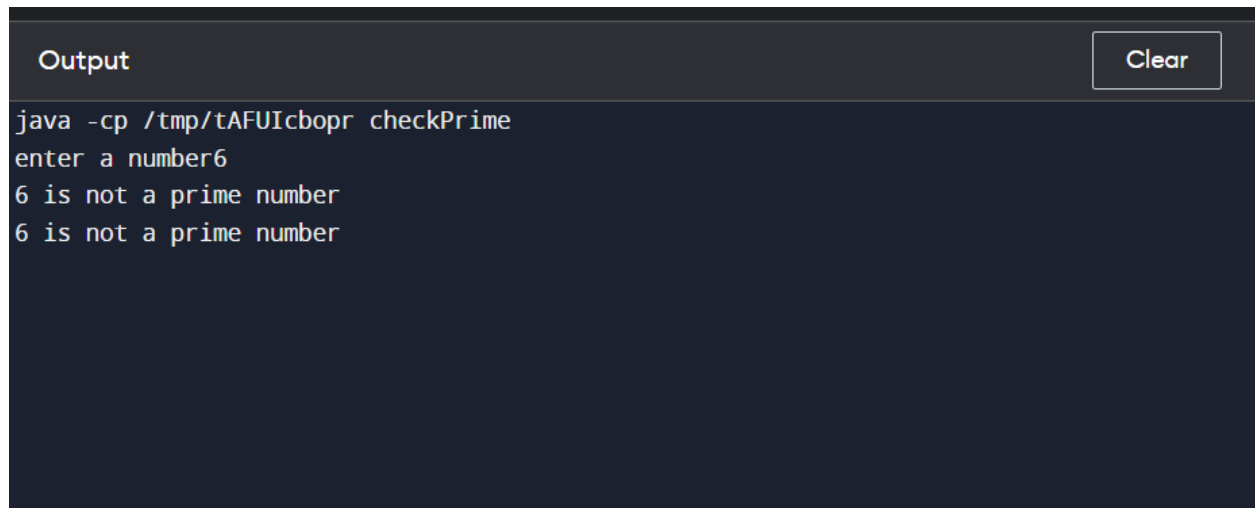
```
import java.util.Scanner;
import java.io.*;

class notPrime extends Exception{
    public String toString(){
        return "Not a Positive number";
    }
}

class checkPrime{
    public static void main(String[] args){
        Scanner sc=new Scanner(System.in);
        System.out.println("enter a number");
        int n=sc.nextInt();
        try{
            if(n>0){
                for(int i=2;i<=n/2;i++){
                    if(n%i==0){
                        System.out.println(n+" is not a prime number");
                    }
                }
            }
        }
        else{
            System.out.println(n+" is a prime number");
        }
    }
}
```

```
        throw new notPrime();
    }
} catch(notPrime np){
    System.out.println(np.toString());
}
}
```

3.Screenshot of excecution:

A screenshot of a Java IDE's output window. The window has a dark background with a title bar that says "Output" and a "Clear" button. The output text is as follows:

```
java -cp /tmp/tAFUIcbopr checkPrime
enter a number6
6 is not a prime number
6 is not a prime number
```

1.Problem definition:

3. Write a Java program to perform the following operations:

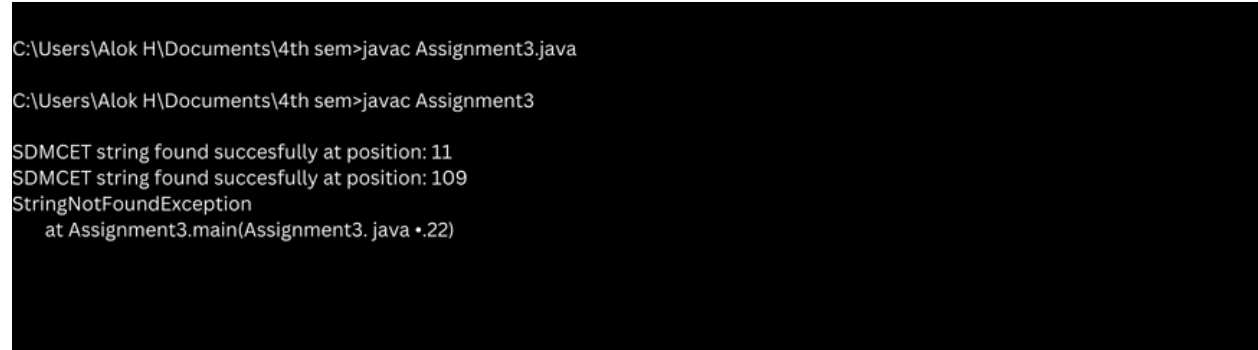
- a) Read a line of text
- b) Search for a sub-string SDMCET (case insensitive search)
- c) If found, then print success message
- d) Otherwise throw an exception SubStringNotFoundException with appropriate message

2.Java Program:

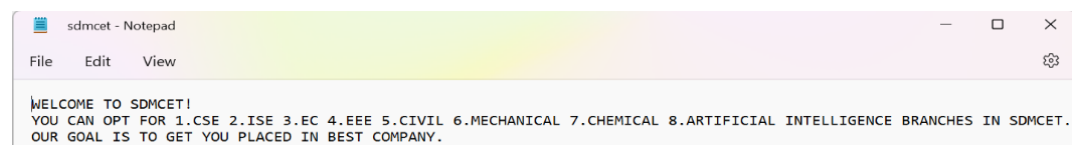
```
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
public class SubString {
    public static void main(String args[]) throws IOException {
        //File f=new File("sdmcet.txt");
        FileReader f=new FileReader("Sdmcet.txt");
        BufferedReader br= new BufferedReader(f);
        String s1="SDMCET";
        String s2="";
        while((s2=br.readLine())!=null) {
            try {
                if(s2.contains(s1)) {
                    System.out.println("SDMCET string found succesfully at position:"+s2.indexOf(s1) );
                }
                else
                    throw new StringNotFoundException("String not found");
            }catch(StringNotFoundException se) {
                se.printStackTrace();
            }
        }
    }
}
```

```
}  
  
class StringNotFoundException extends Exception{  
    private String se;  
    StringNotFoundException(String s){  
        this.se=s;  
    }  
}
```

3.Screenshot of excecution:



```
C:\Users\Alok H\Documents\4th sem>javac Assignment3.java  
  
C:\Users\Alok H\Documents\4th sem>javac Assignment3  
  
SDMCET string found succesfully at position: 11  
SDMCET string found succesfully at position: 109  
StringNotFoundException  
    at Assignment3.main(Assignment3.java •22)
```



1. Problem definition:

4. 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.*;
import java.io.*;

class Assignment4 {
    public static void main(String[] args){
        try{
            FileInputStream fin=new FileInputStream("C:\\Users\\alok\\Documents\\4th
            sem\\Alphabet.txt");

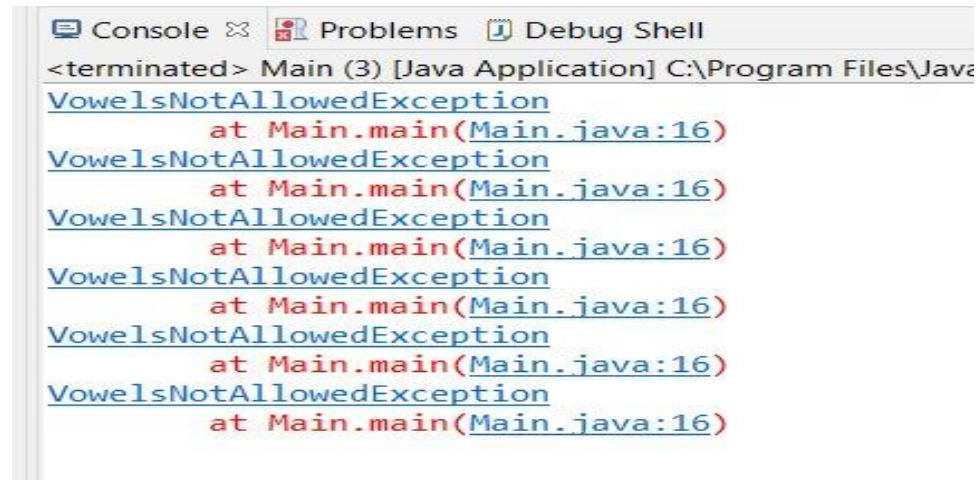
            FileOutputStream fout=new FileOutputStream("C:\\Users\\alok
            \\Documents\\4th sem\\consonant.txt");

            int ch;
            while(ch=fin.read()!=-1){
                if(ch=='a' || ch=='e' || ch=='i' || ch=='o' || ch=='u'){
                    throw new VowelNotAllowedException();
                }
                else
                    fout.write(ch);
            }
        }catch(VowelNotAllowedException e){
            System.out.println(e.toString());
        }
    }
}
```



```
}  
  
class vowelNotAllowedException extends Exception{  
    public String toString(){  
        return "vowels are not allowed";  
    }  
}
```

3.Screenshot of excecution:



The screenshot shows the 'Console' tab of a Java IDE. The output displays a series of stack traces for 'VowelsNotAllowedException'. Each trace starts with the exception name, followed by 'at Main.main(Main.java:16)' repeated seven times. The text is color-coded: exception names are blue and underlined, and the stack trace details are red.

```
<terminated> Main (3) [Java Application] C:\Program Files\Java  
VowelsNotAllowedException  
    at Main.main(Main.java:16)  
VowelsNotAllowedException  
    at Main.main(Main.java:16)  
VowelsNotAllowedException  
    at Main.main(Main.java:16)  
VowelsNotAllowedException  
    at Main.main(Main.java:16)  
VowelsNotAllowedException  
    at Main.main(Main.java:16)  
VowelsNotAllowedException  
    at Main.main(Main.java:16)
```



The screenshot shows a text editor with two tabs: 'Alphabet1.txt' and 'Consonent.txt'. The 'Alphabet1.txt' tab is active, displaying a list of characters with line numbers 1 through 4. The characters are 'h', 'l', 'H', and 'Shn'.

```
Alphabet1.txt  Consonent.txt  
1 hll  
2 H  
3 Shn  
4
```

1.Problem Definition:

5. 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.io.File;
import java.io.FileNotFoundException;
import java.io.FileWriter;
import java.io.IOException;
import java.util.Arrays;
import java.util.Scanner;

class five
{
    private static int arr[];
    public static void main(string []args) throws
    FileNotFoundException,IOException,InterruptedException
    {
        File ipFile = new File("Integers.txt");
        File opFile = new File("SortedIntegers.txt");
        FileWriter opWriter = new FileWriter(opFile);
```

```
Scanner sc = new Scanner(ipFile);

int size = sc.nextInt();

arr = new int[size];

int i = 0;

while (sc.hasNext()) {
    arr[i++] = sc.nextInt();
}

Thread T1 = new Thread() {
    public void run() {
        ThreadSorting(arr, 0, (size / 3) - 1);
    }
};

Thread T2 = new Thread() {
    public void run() {
        ThreadSorting(arr, (size / 3), ((size / 3) * 2) - 1);
    }
};

Thread T3 = new Thread() {
    public void run() {
        ThreadSorting(arr, ((size / 3) * 2), (size - 1));
    }
};

Thread T4 = new Thread() {
    public void run() {
        ThreadSorting(arr, 0, size - 1);
    }
};

T1.start();

T1.join();
```

```
T2.start();
T2.join();
T3.start();
T3.join();
T4.start();
T4.join();
for (int num : arr) {
    opWriter.append(String.valueOf(num) + " ");
}
opWriter.close();
}

public static void ThreadSorting(int arr[], int start, int end) {
    int tempArr[] = new int[end - start + 1];
    int tempIndex = 0;
    for (int i = start; i <= end; i++) {
        tempArr[tempIndex++] = arr[i];
    }
    Arrays.sort(tempArr);
    int index = start;
    for (int n : tempArr) {
        arr[index++] = n;
    }
}
}
```