



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:	2SD20CS081	Name:	Preshita P Desai
------	------------	-------	------------------

1. Problem Definition 1:

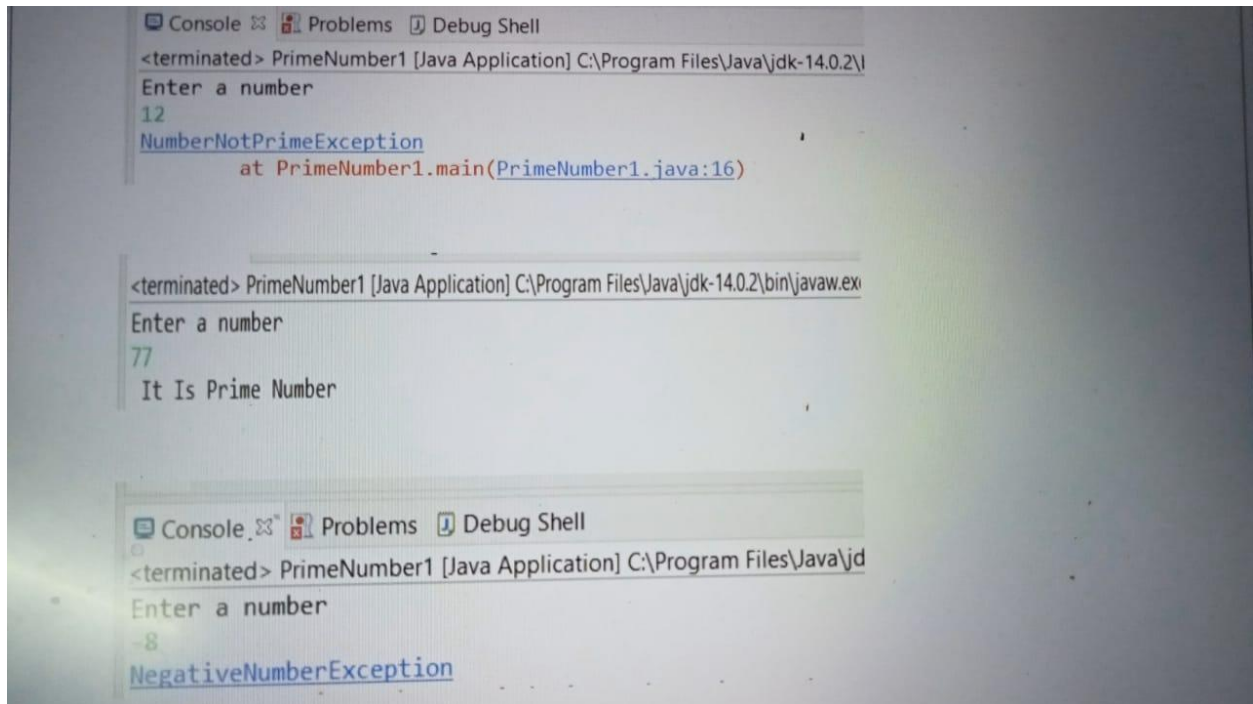
Write a Java program to generate and handle any three built-in exceptions and display appropriate error messages.

2. Java Program:

```
public class MultipleException {  
    public static void main(String args[]) {  
  
        int a=0;  
        int x[]={1,2,3};  
  
        try{  
            int u=x[2]/a;  
        }  
        catch(ArithmeticException e1) {  
            System.out.println("Arithmetic Exception: " +e1);  
        }  
  
        try {  
            String s[]=null;  
            if(s.equals("Java oops")) {  
                System.out.println("equal");  
            }  
        } catch(NullPointerException e2) {  
            System.out.println("Null pointer Exception: " +e2);  
        }  
        try {  
            x[2] =5;  
        }  
        catch(ArrayIndexOutOfBoundsException e3) {
```

```
        System.out.println("Array index out of bound Exception: " +e3);  
    }  
}  
  
}
```

3. Screen Shots of Execution:



The image contains three screenshots of a Java IDE's console window, showing the execution of a program named 'PrimeNumber1'.

The first screenshot shows the program running with the input '12'. It displays the message 'Enter a number' followed by '12'. Then, it throws a `NumberNotPrimeException` at `PrimeNumber1.main(PrimeNumber1.java:16)`.

The second screenshot shows the program running with the input '77'. It displays the message 'Enter a number' followed by '77', and then the output 'It Is Prime Number'.

The third screenshot shows the program running with the input '-8'. It displays the message 'Enter a number' followed by '-8', and then throws a `NegativeNumberException`.

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;

public class Q2{
    public static void main(string[] args)throws numException{
        Scanner sc=new Scanner(System.in);
        System.out.println("enter an integer");
        int n=sc.nextInt();
        if(n<0){
            throw new numException("NegativeNumberNotAllowedException");
        }
        for(int i=2;i<n;i++){
            if(n%i==0)
                throw new numException("NumberNotPrimeException");
        }
        System.out.println("it's prime number");
    }
}

Class numException extends Exception{
    String msg;
    public numException (String msg){
        this.msg=msg;
    }
}
```

```
public String toString(){  
    return "Exception: "+msg;  
}  
}
```


3.Execution:

```
Enter an Integer
34
NumberNotPrimeException
<<< Process finished (PID=8652). (Exit code 0)
===== READY =====
```

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;

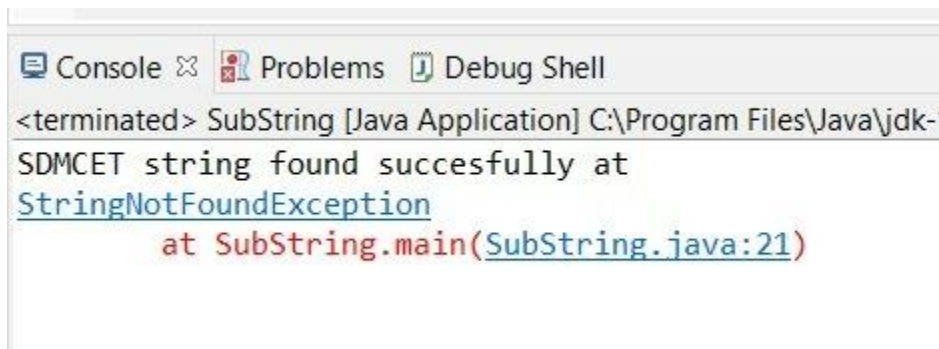
public class SubString {
    public static void main(String args[]) throws IOException {
        FileReader f=new FileReader("Sdmcet.txt");
        BufferedReader br= new BufferedReader(f);
        String s1="SDMCET";
        String s2="";

        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.Execution:



```
<terminated> SubString [Java Application] C:\Program Files\Java\jdk-  
SDMCET string found succesfully at  
StringNotFoundException  
    at SubString.main(SubString.java:21)
```

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.io.*;

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

        FileInputStream fin=new
        //reading text from alphabet.txt file
        FileOutputStream fout=new
        //writing bytes to consonent.txt file


        int s;
        while((s=fin.read())!=-1) {
```



```
try {  
    if(s=='a' || s=='A' || s=='e' || s=='E' || s=='i' ||  
s=='l' || s=='o' || s=='O' || s=='u' || s=='U')  
        throw new VowelsNotAllowedException("Vowels Not Allowed");  
    else  
        fout.write(s);  
} catch (VowelsNotAllowedException e) {  
    e.printStackTrace();  
}  
}  
fin.close();  
fout.close();  
}  
}
```

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

3.Execution:



The screenshot shows an IDE console window with three tabs: 'Console', 'Problems', and 'Debug Shell'. The 'Console' tab is active, displaying the output of a Java application. The output starts with '<terminated> Main (3) [Java Application] C:\Program Files\Java\'. This is followed by seven identical lines of an exception stack trace. Each line consists of the exception name 'VowelsNotAllowedException' in blue, followed by 'at Main.main(Main.java:16)' in red. The lines are separated by blank lines.

```
<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)
VowelsNotAllowedException
    at Main.main(Main.java:16)
```

