



# AOOP Assignment Submission Report

[Submitted as part of CTA Assignment No-1]

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

Submitted by:

USN:	2SD20CS068	Name:	PAVANKUMAR PATTIHAL
------	------------	-------	------------------------

---

## 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:

```
public class Work1 {  
  
    public static void main(String args[])  
  
        {int a = 30;  
  
        int b = 0;  
        String s = null;  
        int d[] = new int[5];  
  
        try {  
  
            int c = a / b;  
  
        } catch (ArithmeticException e)  
  
            { System.out.println("cant devide a number by zero");  
  
            }  
  
        try  
  
            { System.out.println(s.length()  
  
            );  
  
        } catch (NullPointerException f)  
  
            { System.out.println("String is Null");  
  
            }  
  
        try {
```

---

---

```
d[10] = 50;

} catch (ArrayIndexOutOfBoundsException g)

{System.out.println("Array index exceeded");

}

}

}
```

### 3. Screen Shots of Execution:



```
PS E:\A00P> cd "e:\A00P\" ; if ($?) { javac Work1.java } ; if ($?) { java Work1 }
cant devide a number by zero
String is Null
Array index exceeded
PS E:\A00P>
```

### 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:

```
public class Work2 {

    public static void main(String args[])

    {int flag = 0;
```

---

---

```
Scanner s = new Scanner(System.in);

System.out.println("enter the number");


int num = s.nextInt();

try {

if(num < 0) {

    throw new NegativeNumberNotAllowedException();

} else {


        try {

            if (num == 0 || num == 1) {

                throw new NumberNotPrimeException();

            } else {

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

                    {if (num % i == 0) {

                        flag = 1;

                    }

                }

                if (flag == 1) {

                    throw new NumberNotPrimeException();

                } else

                    System.out.println("the given number is prime");

            }

        }

    }

}
```

---

---

```
    } catch (NumberNotPrimeException a)
        {
            System.out.println(a.toString());
        }
    }
    } catch (NegativeNumberNotAllowedException e)
        {
            System.out.println(e.toString());
        }
    }
}
```

```
class NumberNotPrimeException extends Exception
```

```
    {
        NumberNotPrimeException() {
        }

        public String toString() {
            return super.toString() + " : The given number is not prime";
        }
    }
}
```

```
class NegativeNumberNotAllowedException extends Exception
```

```
    {
        NegativeNumberNotAllowedException() {
        }

        public String toString() {
            return super.toString() + " : The given number is negative";
        }
    }
}
```

---

---

### 3. Screen Shots of Execution:

```
PS E:\A00P> javac Work2.java
PS E:\A00P> java Work2
enter the number
4
NumberNotPrimeException : The given number is not prime
PS E:\A00P> java Work2
enter the number
-4
NegativeNumberNotAllowedException : The given number is negative
PS E:\A00P> java Work2
enter the number
7
the given number is prime
```

### 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.Scanner;
```

```
public class Work3 {
```

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

```
    {Scanner s = new Scanner(System.in);
```

```
    System.out.println("enter the string");
```

```
        String string1 = s.nextLine();
```

```
        String string2 = "SDMCET";
```

---

---

```
try {  
  
    boolean b = string1.toLowerCase().contains(string2.toLowerCase());  
  
    if(!b) {  
  
        throw new SubStringNotFoundException();  
    } else  
        System.out.println("substring found");  
    } catch (SubStringNotFoundException e)  
        {System.out.println(e.toString());  
    }  
}
```

```
class SubStringNotFoundException extends Exception  
    {SubStringNotFoundException() {  
  
    }  
    public String toString() {  
        return super.toString() + " : substring not found";  
    }  
}
```

---

---

### 3. Screen Shots of Execution:

```
PS E:\A00P> javac Work3.java
PS E:\A00P> java Work3
enter the string
shashi from sdmcet
substring found
PS E:\A00P> java Work3
enter the string
shashi from dharwad
SubStringNotFoundException : substring not found
```

### 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.io.FileInputStream;

import java.io.FileOutputStream;\

public class Work4 {

    public static void main(String
    args[]){FileInputStream fin = null;

    FileOutputStream fout = null;

    try{
```

---



---

```
fin = new FileInputStream("E:/AOP/Alphabets.txt");
fout = new FileOutputStream("E:/AOP/Consonants.txt");

int ch;

while((ch=fin.read())!=-1){

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

        throw new VowelNotAllowedException();

    else

        fout.write(ch);

}

}

catch(Exception

e){ System.out.println(e.toString());

}

}

}

class VowelNotAllowedException extends Exception

{VowelNotAllowedException() {

}

public String toString() {

return super.toString() + " : Vowels Not Allowed";

}

}
```

### 3. Screen Shots of Execution:

---

```
Alphabets.txt • Consonants.txt
AOOP > Alphabets.txt
1 aeiou
```

```
PS E:\AOOP> cd "e:\AOOP\" ; if ($?) { javac Work4.java } ; if ($?) { java Work4 }
VowelNotAllowedException : Vowels Not Allowed
PS E:\AOOP>
```

```
Alphabets.txt × Consonants.txt
AOOP > Alphabets.txt
1 zxcvbnm
```

```
Alphabets.txt Consonants.txt ×
AOOP > Consonants.txt
1 zxcvbnm
```

## 1. Problem Definition:

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

- Create a file named `Integers.txt` and insert  $n$ -random integers into it
- 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
- 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.util.*;

import java.util.Scanner;

import java.io.*;

public class Work5 {

    public static void main(String[] args)

        {try{

            FileWriter w = new FileWriter("Integer.txt");

            Scanner sc= new Scanner(System.in);

            System.out.println("Enter the value of n Integer to write on a file :"); int

            n = sc.nextInt();

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

                System.out.print("Enter the " + (i + 1) + "to write :"); int

                input = sc.nextInt();

                w.write(input + "\t");

            }w.close();

            int i=0;

            int arr[] = new int[n];

            File file = new File("Integer.txt");

            Scanner read = new Scanner(file);

            while(read.hasNext()){
```

---

```

arr[i++] = Integer.valueOf(read.next());
}
Thread t1= new
    Thread(){public
        void run(){
            Arrays.sort(arr, 0, (arr.length/3));
for (int j = 0; j < (arr.length/3); j++) {
                System.out.println(arr[j]);
            }
        }
};

Thread t2= new
    Thread(){public void
run(){
        Arrays.sort(arr, (arr.length/3), (2*(arr.length/3)));
        for (int j = (arr.length/3); j < (2*(arr.length/3)); j++)
            {System.out.println(arr[j]);
            }
        }
};

Thread t3= new
    Thread(){public
        void run(){
            Arrays.sort(arr, (2*(arr.length/3)),(n-1)); for
            (int j = (2*(arr.length/3)); j < n; j++)

```

---

```
{ System.out.println(arr[j]);  
}
```

```
    }  
};
```

```
Thread t4= new
```

```
Thread(){ public
```

```
void
```

```
run(){ Arrays.sort(a
```

```
rr);
```

```
// Arrays.sort(arr, 0,n);
```

```
StringBuilder s = new StringBuilder(); try{
```

```
FileWriter write =new FileWriter("SortedInteger.txt");
```

```
System.out.println("t4 is printing"); for (int j = 0; j < n;
```

```
j++) {
```

```
s.append(String.valueOf(arr[j]) + "\t");
```

```
}
```

```
write.write(s.toString()); write.close();
```

```
}catch (Exception
```

```
e){System.out.println
```

```
(e);
```

```
}
```

```
}
```

```
};
```

```
t1.start();
```

```
t1.join();
```

```
t2.start();
```

```
t2.join();
```

---

```
t3.start();  
t3.join();
```

```
t4.start();

} catch (Exception e) {

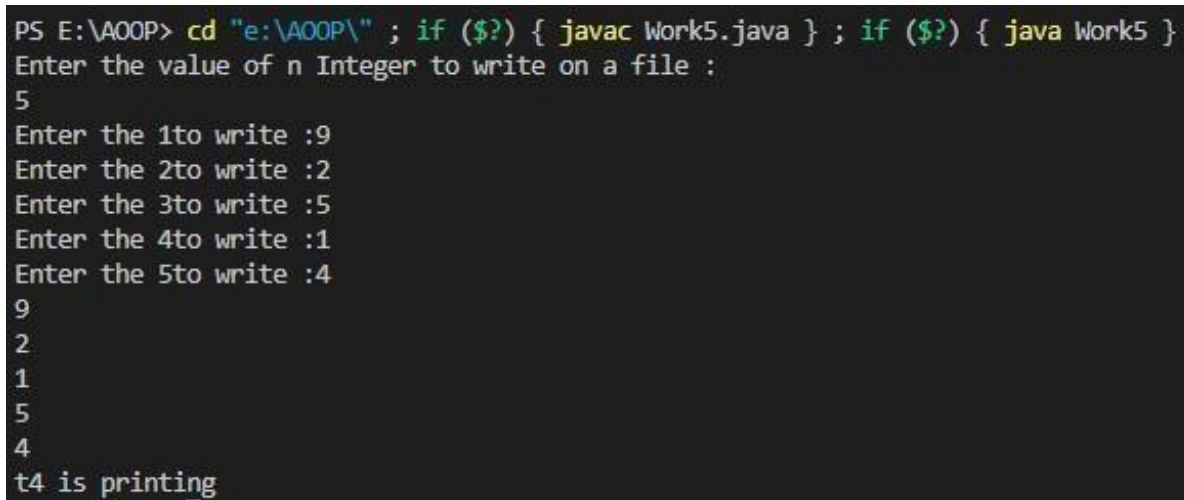
    System.out.println(e);

}

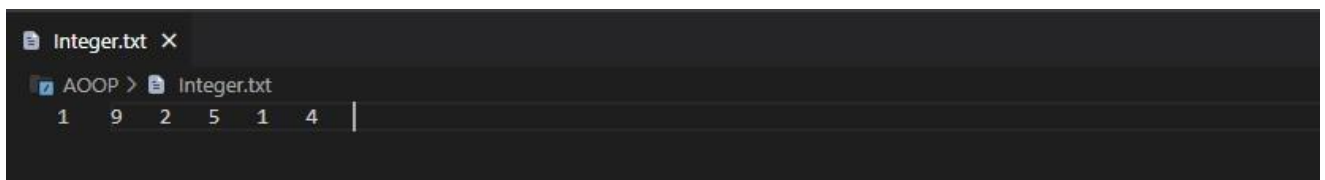
}

}
```

### 3. Screen Shots of Execution:



```
PS E:\AOOP> cd "e:\AOOP\" ; if ($?) { javac Work5.java } ; if ($?) { java Work5 }
Enter the value of n Integer to write on a file :
5
Enter the 1to write :9
Enter the 2to write :2
Enter the 3to write :5
Enter the 4to write :1
Enter the 5to write :4
9
2
1
5
4
t4 is printing
```



```
Integer.txt X
AOOP > Integer.txt
1 9 2 5 1 4 |
```