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:	2SD20CS128	Name:	Yogita B Joshi

1. Problem Definition:

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

```
public class BuiltinException {
	private static String str;

	public static void main(String[] args)
	{
		divideByZero();
		indexOutOfBounds();
		nullPointerException();

	}

	public static void divideByZero() {
		//To generate
		  int num1=10;
		  int num2=0;
		  int res;

		try {
			res=num1/num2;
		}

		catch(ArithmeticException e)
		{
			System.out.println("Divide by zero error");
			System.out.println(e.getMessage());
		}

		}


    public static void indexOutOfBounds() {
	try {
	int num[] = new int[10];
```

```
        num[12] = 10;
        }
        catch(ArrayIndexOutOfBoundsException e)
        {
                System.out.println("\nArray Index is Out Of Bounds");
                System.out.println(e.getMessage());
        }

        }

    public static void nullPointerException() {
        try {
                 str = null;
                System.out.println(str.charAt(0));

       }
        catch (NullPointerException e) {
        System.out.println("\nNull Pointer Exception");
        System.out.println(e.getMessage());
      }
  }
        }
```

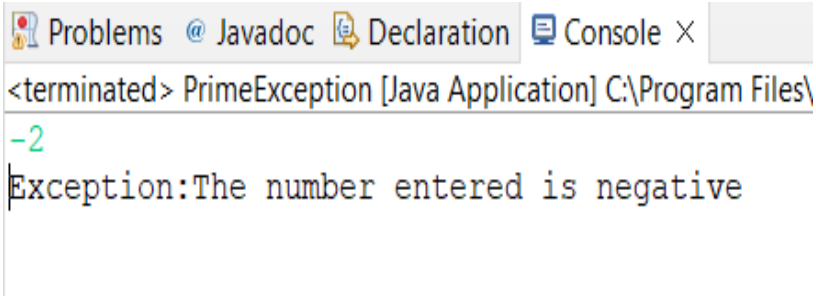
2. Problem Definition:

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.

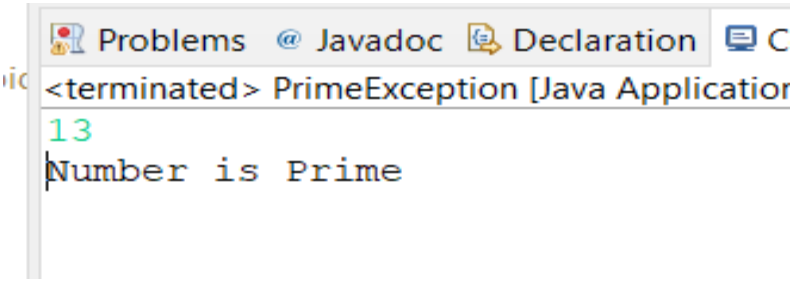
```
import java.util.Scanner;

public class PrimeException {

		public static void main(String[] args) {
		try {
	Scanner sc=new Scanner(System.in);
	int flag=0;



		int num=sc.nextInt();
		if(num<0)
		{
			throw new NegativeNumberNotAllowedException();
		}

			for (int i=1; i<=num; i++)
			{
				if (num%i==0)
			   flag ++;

			}

	if (flag==2)
		  System.out.println("Number is Prime\n");
		else
		{
			throw new NumberNotPrimeException();
		}
	}//end of try

	catch(NumberNotPrimeException ne)
```

```
        {
                System.out.println(ne);
        }

        catch(NegativeNumberNotAllowedException e)
        {
                System.out.println(e);

        }
}
}
                class NegativeNumberNotAllowedException extends Exception{
                        public String toString() {
                                return"Exception:The number entered is negative";

                        }
                }

                class NumberNotPrimeException extends Exception{
                        public String toString() {
                                return"Exception:The number is not prime";
                        }
                }
```

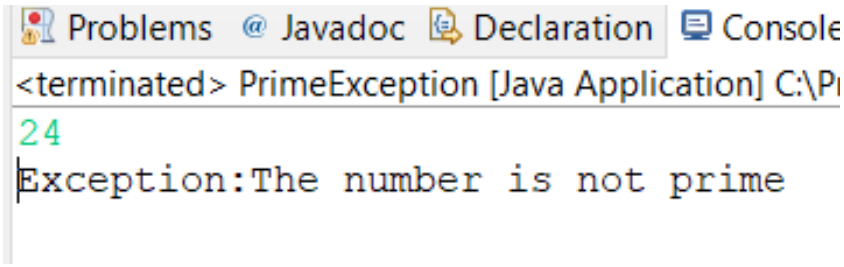
Test case 1:



Test case 2:



Test case 3:



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)

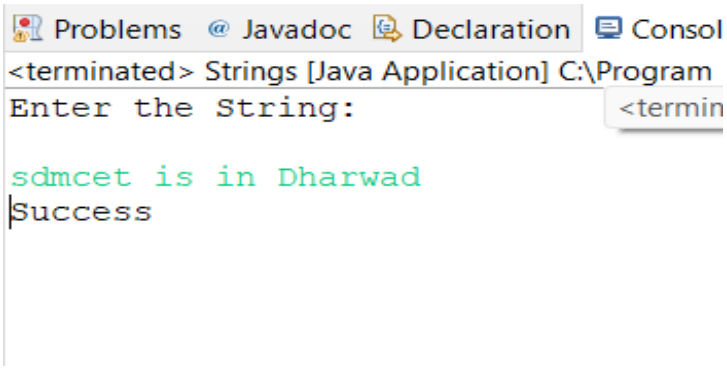
c) If found, then print success message

d) Otherwise throw an exception SubStringNotFoundException with appropriate message

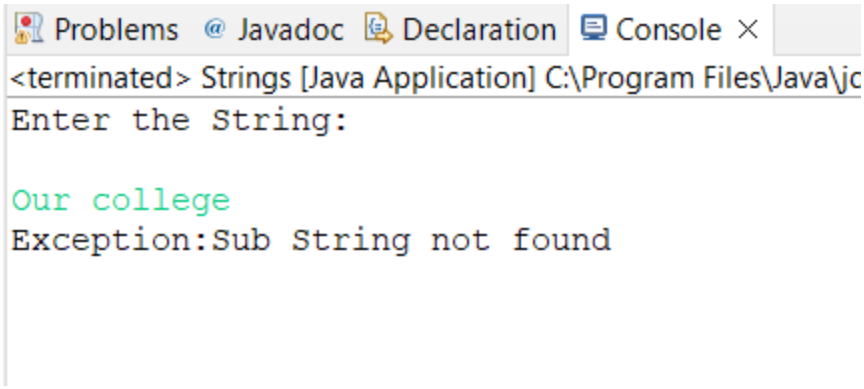
```
import java.util.*;

public class Strings {



        public static void main(String[] args)
        {

                Scanner sc=new Scanner(System.in);

                String str="";
                String str1="";
           System.out.println("Enter the String:\n");

           str=sc.nextLine();


           str1="sdm";
           str=str.toUpperCase();
           str1=str1.toUpperCase();

        try {
        if(str.contains(str1)) {
                System.out.println("Success");
        }
        else
        {
```

```
                throw new SubStringNotFoundException();
        }
        }
        catch(SubStringNotFoundException e)
        {
                System.out.println(e);
        }
        }
}

        class SubStringNotFoundException extends Exception
        {
                public String toString()
                {
                        return"Exception:Sub String not found";
                }
        }
```

Test case 1:



Test case 2:



4. Problem Definition:

Write a Java program to perform the following operations:

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

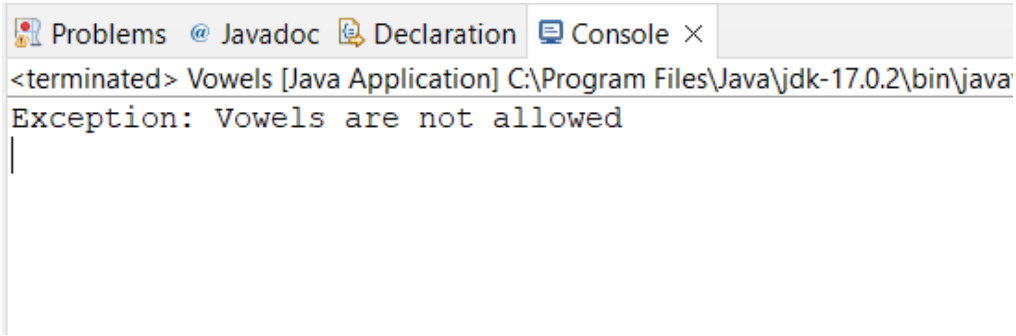
b) Read the file and copy all the consonants into another file named Consonants.txt

c) If vowel is encountered, throw an exception VowelNotAllowedException and continue until end of file

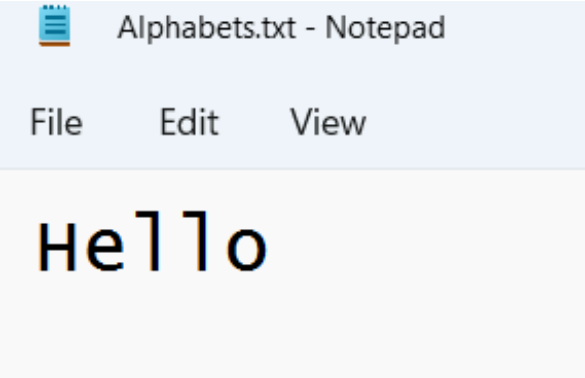
```
import java.util.*;
import java.io.*;

public class Vowels {
	public static void main(String[] args) throws Exception{
		try {
		int flag=0;
		FileInputStream fin=null;
		FileOutputStream fout=null;

			fin=new FileInputStream("C:\\Users\\YOGITA
JOSHI\\OneDrive\\Desktop\\ia 4th\\Vowels\\Alphabets.txt");
			fout=new FileOutputStream("C:\\Users\\YOGITA
JOSHI\\OneDrive\\Desktop\\ia 4th\\Vowels\\Consonants.txt");
			int ch;



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

	if((ch=='a'||ch=='e'||ch=='i'||ch=='o'||ch=='u'||ch=='A'||ch=='E'||ch=='I'||ch=='O'||ch=='U'))
				{
					flag=flag+1;
				}
				else
				{
					fout.write(ch);
				}
```

```
                        }
                        if(flag>0) {
                        throw new VowelNotAllowedException();

                    }
                }

                catch(VowelNotAllowedException e)
                {
                        System.out.println(e);
                }
                catch(IOException ie)
                {
                        System.out.println(ie);
                }

        }
}


class VowelNotAllowedException extends Exception{
        public String toString() {
                return"Exception: Vowels are not allowed";
        }
}
```

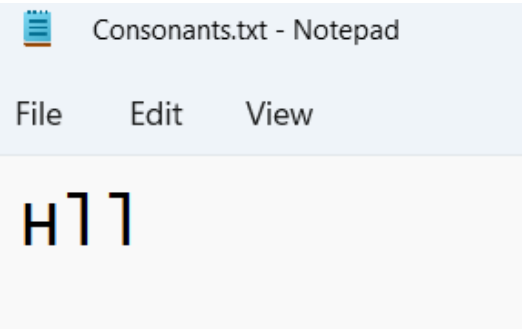
Test case 1:



Alphabets.txt



Consonants.txt



5. Problem Definition

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

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


public class SortThreads {

  public static void main(String[] args) {
    try{
    FileWriter w = new FileWriter("C:\\Users\\YOGITA JOSHI\\OneDrive\\Desktop\\ia 4th\\ThreadSorting\\Integers.txt");
    Scanner sc= new Scanner(System.in);


    System.out.println("Enter total number of values:");
    int n = sc.nextInt();
    for (int i = 0; i < n; i++) {
        System.out.print("Enter the values:" );
        int input = sc.nextInt();
        w.write(input + "\t");
  }w.close();
  int i=0;
  int arr[] = new int[n];
 File file = new File("C:\\Users\\YOGITA JOSHI\\OneDrive\\Desktop\\ia 4th\\ThreadSorting\\Integers.txt");
```

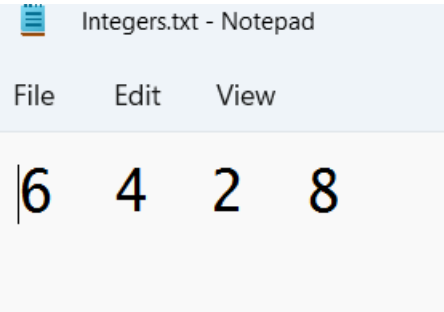
```
Scanner read = new Scanner(file);
while(read.hasNext()){
 arr[i++] = Integer.valueOf(read.next());
}


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

        StringBuilder s = new StringBuilder();
        try{
        FileWriter write    =new FileWriter("C:\\Users\\YOGITA JOSHI\\OneDrive\\Desktop\\ia
4th\\ThreadSorting\\Sorted Integers.txt");
        System.out.println("T4 is in sorting file");
        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);
      }
     }

}
```

Test case :

```
Problems  @ Javadoc  Declaration
<terminated> SortThreads [Java Application]
Enter total number of values:
4
Enter the values:6
Enter the values:4
Enter the values:2
Enter the values:8
6
4
2
8
T4 is in sorting file
```

Integers.txt



Sorted Integers.txt