**CSA09 Programming in Java**

**Day 4 Assignment Questions**

* Create File1.txt file, in which, store more than one line of text. Write a Java program to count the no. of words, characters and lines from the input file File1.txt.

Solution:

import java.io.BufferedReader;

import java.io.FileReader;

import java.io.IOException;

public class FileStats {

public static void main(String[] args) {

String fileName = "File1.txt";

int charCount = 0;

int wordCount = 0;

int lineCount = 0;

try (BufferedReader br = new BufferedReader(new FileReader(fileName))) {

String line;

while ((line = br.readLine()) != null) {

lineCount++;

charCount += line.length();

String[] words = line.split("\\s+");

wordCount += words.length;

}

} catch (IOException e) {

e.printStackTrace();

}

System.out.println("Character count: " + charCount);

System.out.println("Word count: " + wordCount);

System.out.println("Line count: " + lineCount);

}

}

* Create Customer class with deposit() and withdraw() as synchronized methods. Declare AccountNo, AccName and Balance as Instance Variables inside the class. From the main class, Input the amount for withdraw() operation and if requested amount is not available in existing Balance amount, withdraw() method should be temporarily suspended using wait() method until deposit() method receives the input for amount, to be added in the existing Balance amount and then withdraw() would be completed in a successful manner. Develop the above scenario using Synchronization and Inter-Thread Communication.

Solution:

class Customer {

private int accountNo;

private String accName;

private int balance;

public Customer(int accountNo, String accName, int balance) {

this.accountNo = accountNo;

this.accName = accName;

this.balance = balance;

}

public synchronized void deposit(int amount) {

balance += amount;

System.out.println(amount + " deposited successfully");

notify();

}

public synchronized void withdraw(int amount) {

if (balance < amount) {

try {

System.out.println("Not enough balance. Waiting for deposit...");

wait();

} catch (InterruptedException e) {

e.printStackTrace();

}

}

balance -= amount;

System.out.println(amount + " withdrawn successfully");

}

public int getAccountNo() {

return accountNo;

}

public String getAccName() {

return accName;

}

public int getBalance() {

return balance;

}

}

**Debugging**

* Given an integer n, return a string array answer (1-indexed) where:

answer[i] == "FizzBuzz" if i is divisible by 3 and 5.

answer[i] == "Fizz" if i is divisible by 3.

answer[i] == "Buzz" if i is divisible by 5.

answer[i] == i (as a string) if none of the above conditions are true.

Example 1:

Input: n = 3

Output: ["1","2","Fizz"]

Example 2:

Input: n = 5

Output: ["1","2","Fizz","4","Buzz"]

Example 3:

Input: n = 15

Output: ["1","2","Fizz","4","Buzz","Fizz","7","8","Fizz","Buzz","11","Fizz","13","14","FizzBuzz"]

Constraints:

1 <= n <= 104

class Solution {

    vector<string> fizzBuzz(int n) {

    }

}

Solution:

class Solution {

public List<String> fizzBuzz(int n) {

List<String> answer = new ArrayList<>();

for(int i=1; i<=n; i++) {

if(i % 3 == 0 && i % 5 == 0) {

answer.add("FizzBuzz");

} else if(i % 3 == 0) {

answer.add("Fizz");

} else if(i % 5 == 0) {

answer.add("Buzz");

} else {

answer.add(Integer.toString(i));

}

}

return answer;

}

}

* Given two strings s and goal, return true if and only if s can become goal after some number of shifts on s.

A shift on s consists of moving the leftmost character of s to the rightmost position.

For example, if s = "abcde", then it will be "bcdea" after one shift.

Example 1:

Input: s = "abcde", goal = "cdeab"

Output: true

Example 2:

Input: s = "abcde", goal = "abced"

Output: false

Constraints:

1 <= s.length, goal.length <= 100

s and goal consist of lowercase English letters.

class Solution {

    bool rotateString(string s, string goal) {

    }

}

Solution:

class Solution {

public boolean rotateString(String s, String goal) {

if(s.length() != goal.length()) {

return false;

}

String s2 = s + s;

return s2.contains(goal);

}

}

* Program to Find Prime No using multithreading: **Find/Debug error**s in the following code

class PrimeExample implements Runnable

//extends Thread

{

public void run()

{

int i,m=20,flag=1;

for(i=1;i<=m;i++);

{

if(i<=3){

System.out.println(i + " is prime number");

continue;

}

else if(i>3)

{

for(int j=2;j<i;i++)

{

if(i%j==0)

{

flag=0;

break;

}

}

if (flag!= 1)

{

System.out.printn(i + " is not prime number");

flag=1;

}

else

System.out.prinln(i + " is prime number");

}

}

}

}

class prime

{

public static void main(String args[]){

try

{

PrimeExample p1 = new PrimeExample();

Thread t1= new Thread(p2);

t1.start();

}

catch(Exception e)

{

System.out.println(e.getMessage());

}

}

}

Solution:

class PrimeExample implements Runnable {

public void run() {

int i, m = 20, flag;

for (i = 1; i <= m; i++) {

flag = 1;

if (i <= 3) {

System.out.println(i + " is prime number");

continue;

} else if (i > 3) {

for (int j = 2; j < i; j++) {

if (i % j == 0) {

flag = 0;

break;

}

}

if (flag != 1) {

System.out.println(i + " is not prime number");

} else {

System.out.println(i + " is prime number");

}

}

}

}

}

class prime {

public static void main(String args[]) {

try {

PrimeExample p1 = new PrimeExample();

Thread t1 = new Thread(p1);

t1.start();

} catch (Exception e) {

System.out.println(e.getMessage());

}

}

}