Name: Thota Gurutheja Reddy

Regno: 19BCD7034

Lab-9

1. Consider the track line in railways. If multiple trains trying to access the same line it should not allow because it will lead to collision. Now write a java program to schedule the track for the train. Create 3 threads objects for the class Train for 3 trains and schedule the Line by calling the getLine() (Synchronized method on the same Object but only one thread will be able to execute it at a time) in the Line class.

Ans:

class Line

{

synchronized public void getLine()

{

for (int i = 1; i < 4; i++)

{

System.out.println(i);

try

{

Thread.sleep(400);

}

catch (Exception e)

{

System.out.println(e);

}

}

}

}

class Train extends Thread

{

Line line = new Line();

Train(Line line)

{

this.line = line;

}

public void run()

{

line.getLine();

}

}

class Main

{

public static void main(String[] args)

{

Line obj = new Line();

Train train1 = new Train(obj);

Train train2 = new Train(obj);

Train train3 = new Train(obj);

train1.start();

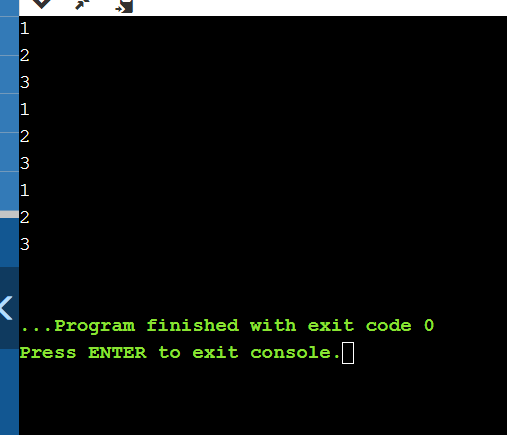
train2.start();

train3.start();

}

}

Output:



2. Create a class serieseven with two methods Series and Printeven. Series method is synchronized to print the series by reading n and generated series should have the value <2000 and Printeven to find the sum of the series generated is even. Create two threads to perform series generation and check whether sum is even or not.

Sample input and output n=2

2

8

18

32

50

100

400

900

1600

5610Sum is EVEN

n=3

3

12

27

48

75

100

400

900

1600

Sum is odd

Ans:

import java.util.\*;

class SeriesSeven extends Thread{

Random r= new Random();

int sum,p;

public void series(){

Scanner sc= new Scanner(System.in);

int n= sc.nextInt();

sum=n;

System.out.println(n);

for(int i=0;i<10;i++){

p=r.nextInt(2000)+n;

System.out.println(p);

sum=sum+p;

}

}

public void printeven(){

System.out.println(sum);

if(sum%2==0){

System.out.println("Sum is even");

}

else {

System.out.println("Sum is odd");

}

}

}

class Guru extends Thread{

SeriesSeven s=new SeriesSeven();

public void run(){

s.series();

s.printeven();

}

}

class Main{

public static void main (String[] args) {

Guru t1 = new Guru();

t1.start();

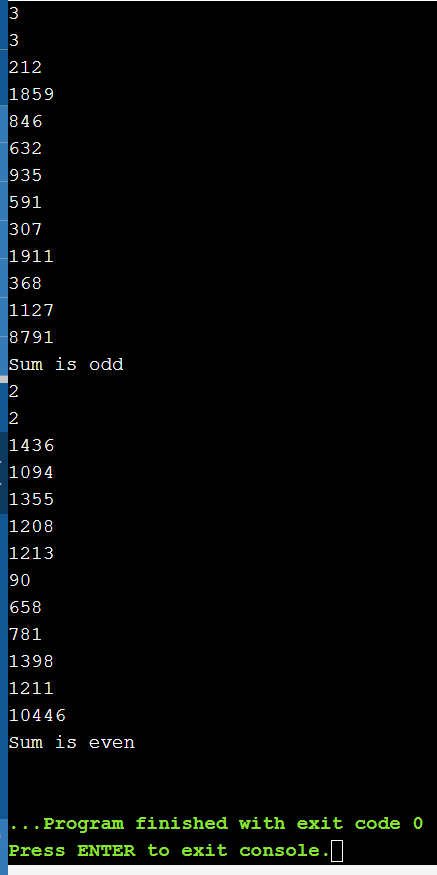
Guru t2 = new Guru();

t2.start();

}

}

Output:



3. a.Create a CeaserCipher class to perform substitution and reverse substitution of characters of a message.

- mEncryption method - substitute a character with another charcter of alphabet.

- mDecryption method - similar to mEncryption method but it performs in reverse.

Each character of message is considered as numeric value with the following mapping:**a-z to 0-25**, respectively.

The mEncryption method replaces each character of the message with another character by using the following formula:(N(ch)+k)%26, where N(ch) means Numeric value of a character 'ch', k means key value 0<=k<=25.

The mDecryption method substitutes each character with the following formula: (N(ch)-k)%26.

Inputs to each method is a message and a key and output is substituted message printed on console character by character.

(Ex: Input to mEncryption is: rama and 25 and output is: qzlz ;

Input to mDecryption is: qzlz and 25 and output is: rama )

Create a TestCeaserCipher class to test mEncryption & mDecryption methods.

b. Jennifer comes with a message "gdhrzfnncanx". She wants to perform reverse substitution using mDecryption method but not aware of key 'k'. To help her, develop a multithreaded program to create separate thread for each possible key 'k' and print all reverse substitutions. Do necessary changes to CeaserCipher class and provide synchronization for threads if the output from threads are mixed.

Ans:

a.

import java.util.Scanner;

class CeaserCipher

{

public static String a = "abcdefghijklmnopqrstuvwxyz";

public static String mEncryption(String p, int key)

{

p = p.toLowerCase();

String c = "";

for (int i = 0; i < p.length(); i++)

{

int cp = a.indexOf(p.charAt(i));

int kv = (key + cp) % 26;

char rv = a.charAt(kv);

c += rv;

}

return c;

}

public static String mDecryption(String c, int key)

{

c = c.toLowerCase();

String p = "";

for (int i = 0; i < c.length(); i++)

{

int cp = a.indexOf(c.charAt(i));

int kv = (cp - key) % 26;

if (kv < 0)

{

kv = a.length() + kv;

}

char rv = a.charAt(kv);

p += rv;

}

return p;

}

public static void main(String[] args)

{

Scanner sc=new Scanner(System.in);

System.out.println("Enter the Message to encrypt");

String m =sc.nextLine();

System.out.println("Enter the Key for encryption");

int key = sc.nextInt();

String c=mEncryption(m, key);

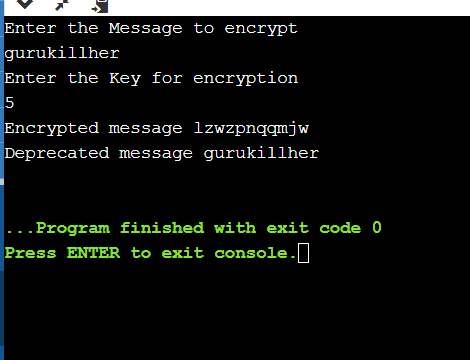
System.out.println("Encrypted message " + c);

System.out.println("Deprecated message " + mDecryption(c, key));

}

}

Output:



b.

public class Main extends Thread

{

public static String m = "gdhrzfnncanx";

public static String a = "abcdefghijklmnopqrstuvwxyz";

static class Decrypt extends Thread

{

int key;

Decrypt(int key)

{

this.key=key;

}

String l = m.toLowerCase();

String p = "";

public void run(){

for (int i = 0; i < l.length(); i++)

{

int cp = a.indexOf(l.charAt(i));

int kv = (cp - key) % 26;

if (kv < 0)

{

kv = a.length() + kv;

}

char r = a.charAt(kv);

p += r;

}

System.out.println("Message for key "+key+ " is " + p);

}

}

public static void main(String[] args)

{

Main.Decrypt [] c=new Main.Decrypt[26];

for(int k=0;k<26;k++){

c[k]=new Main.Decrypt(k);

}

for(int y=0;y<26;y++){

c[y].start();

}

}

}

Output:

