**EX 16**

**Write a java program that correct implements of producer consumer program using the concept of inter thread communication**

// Java program to implement solution of producer

// consumer problem.

import java.util.LinkedList;

public class Threadexample {

public static void main(String[] args)

throws InterruptedException

{

// Object of a class that has both produce()

// and consume() methods

final PC pc = new PC();

// Create producer thread

Thread t1 = new Thread(new Runnable() {

@Override

public void run()

{

try {

pc.produce();

}

catch (InterruptedException e) {

e.printStackTrace();

}

}

});

// Create consumer thread

Thread t2 = new Thread(new Runnable() {

@Override

public void run()

{

try {

pc.consume();

}

catch (InterruptedException e) {

e.printStackTrace();

}

}

});

// Start both threads

t1.start();

t2.start();

// t1 finishes before t2

t1.join();

t2.join();

}

// This class has a list, producer (adds items to list

// and consumer (removes items).

public static class PC {

// Create a list shared by producer and consumer

// Size of list is 2.

LinkedList<Integer> list = new LinkedList<>();

int capacity = 2;

// Function called by producer thread

public void produce() throws InterruptedException

{

int value = 0;

while (true) {

synchronized (this)

{

// producer thread waits while list

// is full

while (list.size() == capacity)

wait();

System.out.println("Producer produced-"

+ value);

// to insert the jobs in the list

list.add(value++);

// notifies the consumer thread that

// now it can start consuming

notify();

// makes the working of program easier

// to understand

Thread.sleep(1000);

}

}

}

// Function called by consumer thread

public void consume() throws InterruptedException

{

while (true) {

synchronized (this)

{

// consumer thread waits while list

// is empty

while (list.size() == 0)

wait();

// to retrieve the first job in the list

int val = list.removeFirst();

System.out.println("Consumer consumed-"

+ val);

// Wake up producer thread

notify();

// and sleep

Thread.sleep(1000);

}

}

}

}

}

**Output:**

Producer produced-0

Producer produced-1

Consumer consumed-0

Consumer consumed-1

Producer produced-2

Consumer consumed-2

**EX 17**

**Write a java program to find the maximum value from the given type of elements using a generic function.**

class MyClass<T extends Comparable<T>>

{

T[] vals;

MyClass(T[] o)

{

vals = o;

}

public T min()

{

T v = vals[0];

for(int i=1; i < vals.length; i++)

if(vals[i].compareTo(v) < 0)

v = vals[i];

return v;

}

public T max()

{

T v = vals[0];

for(int i=1; i < vals.length;i++)

if(vals[i].compareTo(v) > 0)

v = vals[i];

return v;

}

}

class gendemo

{

public static void main(String args[])

{

int i;

Integer inums[]={10,2,5,4,6,1};

Character chs[]={'v','p','s','a','n','h'};

Double d[]={20.2,45.4,71.6,88.3,54.6,10.4};

MyClass<Integer> iob = new MyClass<Integer>(inums);

MyClass<Character> cob = new MyClass<Character>(chs);

MyClass<Double>dob = new MyClass<Double>(d);

System.out.println("Max value in inums: " + iob.max());

System.out.println("Min value in inums: " + iob.min());

System.out.println("Max value in chs: " + cob.max());

System.out.println("Min value in chs: " + cob.min());

System.out.println("Max value in chs: " + dob.max());

System.out.println("Min value in chs: " + dob.min());

}

}

**EX 18**

**Write a java program that simulates a traffic light. The program lets the user select one of three lights: Red, Yellow or Green with radio buttons. On selecting a button an appropriate message with“STOP “or “READY” or “GO” should appear above the buttons in selected color. Initially, there is no message shown.**

import javax.swing.\*;

import javax.swing.event.\*;

import java.awt.\*;

import java.awt.event.\*;

class TrafficLightSimulator extends JFrame implements ItemListener {

JLabel lbl1, lbl2;

JPanel nPanel, cPanel;

CheckboxGroup cbg;

public TrafficLightSimulator() {

setTitle("Traffic Light Simulator");

setSize(600,400);

setLayout(new GridLayout(2, 1));

nPanel = new JPanel(new FlowLayout());

cPanel = new JPanel(new FlowLayout());

lbl1 = new JLabel();

Font font = new Font("Verdana", Font.BOLD, 70);

lbl1.setFont(font);

nPanel.add(lbl1);

add(nPanel);

Font fontR = new Font("Verdana", Font.BOLD, 20);

lbl2 = new JLabel("Select Lights");

lbl2.setFont(fontR);

cPanel.add(lbl2);

cbg = new CheckboxGroup();

Checkbox rbn1 = new Checkbox("Red Light", cbg, false);

rbn1.setBackground(Color.RED);

rbn1.setFont(fontR);

cPanel.add(rbn1);

rbn1.addItemListener(this);

Checkbox rbn2 = new Checkbox("Orange Light", cbg, false);

rbn2.setBackground(Color.ORANGE);

rbn2.setFont(fontR);

cPanel.add(rbn2);

rbn2.addItemListener(this);

Checkbox rbn3 = new Checkbox("Green Light", cbg, false);

rbn3.setBackground(Color.GREEN);

rbn3.setFont(fontR);

cPanel.add(rbn3);

rbn3.addItemListener(this);

add(cPanel);

setVisible(true);

// to close the main window

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

}

// To read selected item

public void itemStateChanged(ItemEvent i) {

Checkbox chk = cbg.getSelectedCheckbox();

String str=chk.getLabel();

char choice=str.charAt(0);

switch (choice) {

case 'R':lbl1.setText("STOP");

lbl1.setForeground(Color.RED);

break;

case 'O':lbl1.setText("READY");

lbl1.setForeground(Color.ORANGE);

break;

case 'G':lbl1.setText("GO");

lbl1.setForeground(Color.GREEN);

break;

}

}

// main method

public static void main(String[] args) {

new TrafficLightSimulator();

}

}