**Pseudo code of solving Producer consumer problem**

The Program contains 4 main classes {producerConcumerIMP, Producer, Consumer and main Class)

**producerConcumerIMP Class**

**Attributes:**

\*capacity (number of spaces that Queue can hold)

\* Queue (space that produced and consumed)

\* Random

\* lock (loch support synchronization)

\* BufferNotFull(Boolean check if the buffer is full or not)

\* BufferNotEmpty(Boolean check if the buffer is empty or not)

\* number

\* numofTickets (num of tickets that are valid)

**Method:**

Put()

While(number is 10 ){stay in infinity loop until provide a space }

Try:

while( Queue size is equal capacity)

Print( the current threadis waiting )

End while

Number = number+1

isAdded = queue.offer(number)

if(isAdded is True)

print(current thread add to the queue )

BufferNotFull.signalAll (check that there is space in queue)

End if

End Try

Unlock the critical section

get()

Try:

while( Queue size is 0)

Print( the current thread is waiting )

End while

value = the front element in the queue

if(value is not NULL)

print(current thread comsume a value from queue )

while( value is 10)

close the system

end while

End if

End Try

Unlock the critical section

**producer Class**

**Attributes:**

\*pc (the Queue that can produce in it)

**Method:**

Run()

While(true)

Try:

Pc.put() (Run out the method put() in producerConcumerIMP class)

Sleep the thread for about 10 seconds

End Try

Catch:

Print any error exist during running

**Consumer Class**

**Attributes:**

\*pc (the Queue that can produce in it)

**Method:**

Run()

While(true)

Try:

Pc.get() (Run out the method get() in producerConcumerIMP class)

Sleep the thread for about 10 seconds

End Try

Catch:

Printany error exist during running

**Main Class**

**Attributes:**

\*object from producerConcumerIMP

\*2object from producer

\*2object from consumer

set first producer object with name : producer 1

set first consumer object with name : consumer 1

set second producer object with name : producer 2

set second consumer object with name : consumer 2

Examples of Deadlock

public static void main(String[] args) {

CustomBlockingQueue blockingQueue = new CustomBlockingQueue();

new Thread(new Producer(blockingQueue)).start();

new Thread(new Consumer(blockingQueue)).start();

}

}

@SuppressWarnings("serial")

class CustomBlockingQueue extends LinkedList <Object> {

private static final int MAX\_SIZE = 10;

private Semaphore mutex = new Semaphore(1);

private Semaphore fillCount = new Semaphore(0);

private Semaphore emptyCount = new Semaphore(MAX\_SIZE);

@Override

public boolean offer(Object e) {

try {

mutex.acquire();

} catch (InterruptedException e2) {

e2.printStackTrace();

}

boolean result = super.offer(e);

System.out.println("offer " + size());

try {

fillCount.release();

emptyCount.acquire();

mutex.release(); }

catch (InterruptedException e1)

{ e1.printStackTrace(); }

return result;

}

@Override

public Object poll() {

try {

mutex.acquire();

} catch (InterruptedException e2) {

e2.printStackTrace(); }

Object result = super.poll();

System.out.println("poll " + size());

try {

emptyCount.release();

fillCount.acquire();

mutex.release();

} catch (InterruptedException e) {

e.printStackTrace();

}

return result;

}

class Producer implements Runnable {

private CustomBlockingQueue blockingQueue;

private Random random = new Random();

public Producer(CustomBlockingQueue blockingQueue) {

this.blockingQueue = blockingQueue; }

@Override

public void run() {

while (!Thread.currentThread().isInterrupted()) {

try {

TimeUnit.SECONDS.sleep(random.nextInt(2));

blockingQueue.offer(new Object());

} catch (InterruptedException e) {

e.printStackTrace(); }

}

}

}

class Consumer implements Runnable {

private CustomBlockingQueue blockingQueue;

private Random random = new Random();

public Consumer(CustomBlockingQueue blockingQueue) {

this.blockingQueue = blockingQueue;

}

@Override

public void run() {

while (!Thread.currentThread().isInterrupted()) {

try {

TimeUnit.SECONDS.sleep(random.nextInt(4));

blockingQueue.poll();

} catch (InterruptedException e) {

e.printStackTrace();

}

}

How did solve deadlock

import java.util.ArrayDeque;

import java.util.Queue;

public class Prod\_consumer {

static Queue q = new ArrayDeque(10);

static class Producer implements Runnable {

public void run() {

while (true) {

if (q.size() == 10) {

synchronized (q) {

try {

System.out.println("Q is full so waiting");

q.wait();

} catch (InterruptedException ex) {

ex.printStackTrace(); }

}

}

synchronized (q) {

String st = System.currentTimeMillis() + "";

q.add(st);

q.notifyAll();

}

}

}

}

static class Consumer implements Runnable {

public void run() {

while (true) {

if (q.isEmpty()) {

synchronized(q) {

try {

System.out.println("Q is empty so waiting ");

q.wait();

}catch(InterruptedException ie) {

ie.printStackTrace(); }

}

}

synchronized(q) {

System.out.println(q.remove());

q.notifyAll();

}

public static void main(String args[]) {

Thread consumer = new Thread(new Consumer());

Thread consumer2 = new Thread(new Consumer());

Thread producer = new Thread(new Producer());

producer.start();

consumer.start();

consumer2.start();

}

}

static class Producer implements Runnable {

public void run() {

while (true) {

synchronized (q) {

if (q.size() < 10) {

String st = System.currentTimeMillis() + "";

q.add(st);

q.notifyAll();

} else {

try {

System.out.println("Q is full so waiting");

q.wait();

} catch (InterruptedException ex) {

ex.printStackTrace();

}

}

}

}

Examples of starvation

public class InterThreadCommunication\_Producer\_Consumer {

static Queue queue = new LinkedList<>();

static int size = 4;

public static void produce() throws InterruptedException {

int value = 0;

while(true) {

synchronized (queue) {

while(queue.size() >= size) { queue.wait(); }

queue.add(value);

System.out.println("Produced" + value);

value++;

queue.notify();

Thread.sleep(1000);

}

} }

public static void consume() throws InterruptedException {

while(true) {

synchronized (queue) {

while(queue.isEmpty()) {

queue.wait();

}

int value = queue.poll();

System.out.println("Consume" + value);

queue.notify();

Thread.sleep(1000);

}

}

}

public static void main(String[] args) throws InterruptedException {

Thread producerThread = new Thread(new Runnable() {

@Override

public void run() {

try {

produce();

} catch (InterruptedException e) {

e.printStackTrace();

}

}

}

Thread consumerThread = new Thread(new Runnable() {

@Override

public void run() {

try {

consume();

} catch (InterruptedException e) {

e.printStackTrace();

}

}

}

producerThread.start();

consumerThread.start();

producerThread.join();

consumerThread.join();

}

}

Then

Time 0 1

Thread 1: Lock(A) Wait-Lock(B)

Thread 2: Lock(B) Wait-Lock(A)

How did solve starvation

class Producer implements Runnable{

private static ArrayList arrTasks = new ArrayList();

void getTasks()

{

Task t = getTask(); // get Tasks from a producer specific recordset.

synchronized (arrTasks) {

arrTasks.add(t);

arrTasks.notify();

}

}

void run() {

while (true) {

Task t = null;

synchronized (arrTasks) {

if (arrTasks.size() == 0)

arrTasks.wait();

if (arrTasks.size() > 0)

t = arrTasks.remove(0);

}

if (t != null)

processTask(t);

if (mExit)

break;

}

}

}