**Operating Systems and Concurrency**

**Student Name:** Raphael Salaja

**AIT Student number:** A00269349

**Assignment:** Assignment 4

**Date of Submission:** 3/11/2021

**FORM A1**

# STUDENT PLAGIARISM DISCLAIMER FORM

****

## PLAGIARISM DISCLAIMER

STUDENT NAME: Raphael Salaja

STUDENT NUMBER: A00269349

PROGRAMME: BSc (Hons) in Software Design with Virtual Reality and Gaming

YEAR: 3

MODULE: Operating Systems and Concurrency

LECTURER: Thiago Braga Rodrigues

ASSIGNMENT TITLE: Lab 4 Assignment

DUE DATE: 10 November2021

DATE SUBMITTED: 11 November 2021

ADDITIONAL INFORMATION:

I understand that plagiarism is a serious academic offence, and that AIT deals with it according to the AIT Policy on Plagiarism.

I have read and understand the AIT Policy on Plagiarism and I agree to the requirements set out therein in relation to plagiarism and referencing. I confirm that I have referenced and acknowledged properly all sources used in preparation of this assignment. I understand that if I plagiarise, or if I assist others in doing so, that I will be subject to investigation as outlined in the AIT Policy on Plagiarism.

I understand and agree that plagiarism detection software may be used on my assignment. I declare that, except where appropriately referenced, this assignment is entirely my own work based on my personal study/or research. I further declare that I have not engaged the services of another to either assist in, or complete this assignment.

## Signed:Raphael Salaja

**Dated: 3 November 2021**

# Contents

[STUDENT PLAGIARISM DISCLAIMER FORM 2](#_Toc86839418)

[PLAGIARISM DISCLAIMER 2](#_Toc86839419)

[Signed:Raphael Salaja 2](#_Toc86839420)

[Contents 3](#_Toc86839421)

[Wait Notify Test 4](#_Toc86839422)

[Waiter 4](#_Toc86839423)

[Notifier 4](#_Toc86839424)

[Message 5](#_Toc86839425)

[WaitNotifyTest 5](#_Toc86839426)

[Output 6](#_Toc86839427)

# Wait Notify Test

## Waiter

public class Waiter implements *Runnable* {  
 private Message msg;  
 public Waiter(Message m) {this.msg = m;}  
 @Override  
 public void run() {  
 String name = Thread.*currentThread*().getName();  
 synchronized (msg) {  
 try {  
 System.*out*.println(name + " waiting to get notified at time: " + System.*currentTimeMillis*());  
 msg.wait();  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 System.*out*.println(name + " waiter thread got notified at time: " + System.*currentTimeMillis*());  
 //process the message now  
 System.*out*.println(name + " processed: " + msg.getMsg());  
 }  
 }  
}

## Notifier

Note with notify it only produces the result for one thread, with notify all it returns all threads

*/\*\*  
 \* NOTIFIER  
 \** ***@author*** *Raphael Salaja  
 \** ***@version*** *3/11/21  
 \*/*

public class Notifier implements *Runnable* {  
 private Message msg;  
 public Notifier(Message msg) {this.msg = msg;}  
  
 @Override  
 public void run() {  
 String name = Thread.*currentThread*().getName();  
 System.*out*.println(name + " started");  
 try {  
 Thread.*sleep*(1000);  
 synchronized (msg) {  
 //Set message and notify  
 msg.setMsg("notifier " + name + " work done");  
 msg.notifyAll();  
 }  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 }  
}

## Message

*/\*\*  
 \* MESSAGE  
 \** ***@author*** *Raphael Salaja  
 \** ***@version*** *3/11/21  
 \*/*

public class Message {  
 private String msg;  
 public Message(String str) {this.msg = str;}  
 public String getMsg() {return msg;}  
 public void setMsg(String str) {this.msg = str;}  
}

## WaitNotifyTest

*/\*\*  
 \* WAIT NOTIFY TEST  
 \** ***@author*** *Raphael Salaja  
 \** ***@version*** *3/11/21  
 \*/*

public class WaitNotifyTest {  
  
 public static void main(String[] args) {  
 // CREATE A Message(), TWO Waiters(msg), AND A Notifier(msg)  
 Message message = new Message(" ");  
 Waiter waiter\_1 = new Waiter(message);  
 Waiter waiter\_2 = new Waiter(message);  
 Notifier notifier = new Notifier(message);  
  
 // START WAITERS AND NOTIFIER  
 *Runnable* run\_waiter\_1 = waiter\_1;  
 *Runnable* run\_waiter\_2 = waiter\_2;  
 *Runnable* run\_notifier\_1 = notifier;  
  
 Thread thread\_run\_waiter\_1 = new Thread(run\_waiter\_1, "Waiter 1");  
 Thread thread\_run\_waiter\_2 = new Thread(run\_waiter\_2, "Waiter 2");  
 Thread thread\_run\_notifier\_1 = new Thread(run\_notifier\_1, "Notifier 1");  
  
 thread\_run\_waiter\_1.start();  
 thread\_run\_waiter\_2.start();  
 thread\_run\_notifier\_1.start();  
  
 // PRINT MESSAGE "ALL THE THREADS ARE STARTED"  
 System.*out*.println("ALL THE THREADS ARE STARTED");  
 }  
}

## Output

