UNIVERSITY OF WESTMINSTER SCHOOL OF COMPUTER SCIENCE & ENGINEERING

Module Title:

Concurrent Programming

Module Code: 6SENG002W / 6SENG004C

Module Leader: Paul Howells Exam Period: January 2019 Time Allowed: 2 Hours

INSTRUCTIONS FOR CANDIDATES

You are advised (but not required) to spend the first ten minutes of the examination reading the questions and planning how you will answer those you have selected.

Answer ANY THREE questions.

Each question is worth 33 marks.

Only the THREE questions with the HIGHEST MARKS will count towards the FINAL MARK for the EXAM.

THIS PAPER MUST NOT BE TAKEN OUT OF THE EXAMINATION ROOM

DO NOT TURN OVER THIS PAGE UNTIL THE INVIGILATOR INSTRUCTS YOU TO DO SO

© UNIVERSITY OF WESTMINSTER, 2019

School of Computer Science and Engineering *M*odule: Concurrent Programming Module Code: 6SENG002W, 6SENG004C

**Question 1**

The following Finite State Process (FSP) processes are used to model a Drinks Vending Machine (DVM), a tea drinking customer (TeaCustomer), a coffee drink ing customer (LatteCustomer) and the complete system (System).

DVM = ( teaButton

-> pay70p -> deliverTea

-> takeTea

-> DVM

latteButton -> pay90p -> deliverLatte -> takeLatte -> DVM ).

TeaCustomer = ( teaButton -> pay70p -> takeTea

-> drinkTea -> TeaCustomer ).

LatteCustomer = ( latteButton -> pay90p -> takeLatte

-> drinkLatte -> LatteCustomer ).

|| System = ( DVM || TeaCustomer || LatteCustomer ).

Given the above FSP process definitions:

(a)

(i)

[3 marks]

(ii)

State the *alphabet*s of the three processes: DVM, TeaCustomer and LatteCustomer. Using your answer to part (i) draw the A*lphabet* diagram for the composite process System. Based on your Alphabet diagram for System, state for each action whether it is s*ynchronous* or *asynchronous* and the processes that perform it.

[7 marks]

(iii)

[5 marks]

(b)

Give the *trace tre*e and the *Labelled Transition System* (LTS) graph for the following processes:

(0) (ii)

LatteCustomer DVM

[6 marks] [8 marks]

(c) In the context of concurrent programming what does *deadlock* mean*?*

How is deadlock represented in the FSP language*?* How is a deadlocked state represented in a Labelled Transition System (LTS) graph?

[4 marks] [TOTAL 33]

© University of Westminster, 2019

Page 1 of 7

School of Computer Science and Engineering Module: Concurrent Programming M*o*dule Code: 6SENGOO2W, 6SENG0040

Question 2

A free music concert is to be held in a Concert Hall, the specification of the system is as follows:

• The Concert Hall has a seating capacity of 100.

. The Concert Hall has one Entrance door and one Exit door.

. The Concert Hall must keep track of how many people are in it, so there is

one Doorman controlling the Entrance and a second Doorman controlling the Exit.

• The two doormen record when someone enters or leaves the Concert Hall

by using a shared counter.

• The Entry doorman process called EntryDoorman, who reads the value of

the counter, add 1 and then updates its value when someone enters the Concert Hall.

• The Exit doorman process called ExitDoorman, who reads the value of

the counter, subtracts 1 and then updates its value when someone leaves the Concert Hall.

To ensure that an accurate record of the number of people in the Concert Hall is maintained, the two doormen must have mutually exclusively access to the shared counter.

(a) Using the Finite State Process (FSP) language define the three processes

to model the shared counter, Entry Doorman and Exit Doorman.

[25 marks]

(b)

Using your three processes define a composite process that models the Concert Hall system.

[4 marks]

(c)

Briefly describe how you have ensured that the two processes modelling the two Doormen have *mutually exclusive* access to the counter.

[4 marks] (TOTAL 33]

© University of Westminster, 2019

Page 2 of *7*

School of Computer Science and Engineering Module: Concurrent Programming Module Code: 6SENG002W, 6SENG004C

Question 3

(a)

The Java programming language facilitates concurrent programming by providing the *threading* mechanism. Describe and define the general con cept of a *thread.*

(5 marks]

(b)

The life-cycle of a Java thread spans its creation, execution and final ter mination. Describe all of the logical states it may be placed in during its life-cycle. In addition describe how and why a thread enters and exits these possible states. Your answer should be illustrated by a diagram and where appropriate, you should refer to the program code given in Appendix B.

(18 marks]

(c) With reference to the program given in Appendix A, briefly describe the

Java scheduling algorithm. Explain how the Java scheduler would sched ule the Racer threads, and give an example of the output that could be produced.

[10 marks] (TOTAL 33]

Question 4

Appendix B contains a Java program which provides a simple simulation of send ing an SM*S text messag*e to a mobile phone.

(a)

Briefly describe the main features of the *monitor* concurrent programming language mechanism, as described by C.A.R. Hoare in his "classic" paper published in 1974.

[5 marks]

(b)

The Java language designers choose to incorporate the *monitor* mechanism in Java, in an attempt to help programmers ensure the safe sharing of resource in multi-threaded Java programs. Describe in detail how the *monitor* mechanism has been implemented in Java. Your answer should be illustrated by reference to the program code given in Appendix B.

[16 marks]

(c) Describe the main differences between Java's version of the *monitor* mech

anism and the "classic" version as described by C.A.R. Hoare.

[5 marks]

(d) With reference to the program given in Appendix B, describe in detail

the sequence of states of the object jillsphone and the threads steve and jill during its execution; assuming that jill calls jillsphone's readtext() method before steve calls its sendtext() method.

17 marks] [TOTAL 33]

© University of Westminster, 2019

Page 3 of 7

School of Computer Science and Engineering Module: Concurrent Programming Module Code: 6SENG002W, 6SENG004C

Question 5

(a) Describe the features of the s*emaphor*e concurrent programming mecha

nism.

[9 marks)

(b) What are the advantages and disadvantages of using semaphores? Explain

wh*y monitors* are generally considered to be "better” than semaphores?

[7 marks]

(c)

(i)

Assuming that you have available a Java Semaphore class, that implements a semaphore. Give suitable Java code fragments to illustrate how semaphores can be used to achieve *mutual exclusion* of a critical section by two Java threads. With reference to your code given in answer to part (i); explain how you have used the semaphore mechanism to achieve mutual exclusion of the critical section.

[14 marks]

[3 marks] [TOTAL 33]

© University of Westminster, 2019

Page 4 of 7

School of Computer Science and Engineering Module: Concurrent Programming Module Code: 6SENG002W, 6SENG0040

**Appendix A**

consists of tw*o* classes: Racer and

The following thread racer program RaceStarter

1.

class Racer extends Thread

malay

Racer(int id)

{

super( "Racer(" + id + "]" );

public void run()

{

for ( int i = 1; i < 40 ; i++)

if ( i % 10 == 0 )

System.out.println( getName() + ", i = " + i); yield();

wwww

class RaceStarter

public static void main( String args[] )

Racer[] racer = new Racer [4] ;

{

for ( int i = 0; i <4 ; i++)

racer[i] = new Racer(i) ;

racer [0].setPriority *(*7) ; racer[3].setPriority(2) ;

{

for (int i = 0; i < 4 ; i++)

racer[i].start();

*}*

37

}

© University of Westminster, 2019

Page 5 of *7*

School of Computer Science and Engineering Module: Concurrent Programming Module Code: 6SENG002W, 6SENG0040

**Appendix B**

The following mobile phone texting program comprises four classes: Texter, Recipient, MobilePhone and SMS.

2

class Texter extends Thread {

private final MobilePhone friendsphone ;

public Texter( MobilePhone phone )

{

friendsphone = phone ;

public void run()

String myTxt = new String("see u @ uni. Spk 18r."); friendsphone. sendtext( myTxt ) ;

class Recipient extends Thread

private final MobilePhone myphone ;

public Recipient( MobilePhone phone )

myphone = phone ;

public void run()

String textmessage = myphone.readtext();

30

}

[Continued Overleaf]

© University of Westminster, 2019

Page 6 of 7

School of Computer Science and Engineering Module: Concurrent Programming Module Code: 6SENG002W, 6SENG0040

31 32

class MobilePhone

{

private String textmessage = null ; private boolean got\_message = false ;

public synchronized void sendtext ( String message )

while ( got\_message ) {

try {

wait(); } catch (InterruptedException e) { }

textmessage = message ; got\_message = true; notify*(*);

public synchronized String readtext()

while ( !got\_message ) {

try {

wait(); } catch (InterruptedException e) { }

got\_message = false; notify(); return textmessage ;

class SMS

public static void main( String args[] ) {

MobilePhone jillsphone = new MobilePhone(); Texter

steve = new Texter( jillsphone ); Recipient

jill = new Recipient( jillsphone );

steve.start(); jill.start();

71

}

© University of Westminster, 2019

Page 7 of 7