**Students: Yong Chen, Jamie Pham**

**Choice: Fish Counter**

Customers continuously, concurrently arrive and take a number. When one of the employees is free they call the next number. Customers require a variable (random) amount of service time.

**Task B**

1. What is the best action a system can take when deadlock is detected?

A deadlock usually occurs when a process that is waiting for one resource is still using another resource which is being waited by another process. To keep it simple, suppose we have these:

Process A

Process B

Resource X

Suppose Process A tries to access Resource X and finishes processing but doesn't unlock the resource. Once Process B tries to access Resource X, it will wait until Process A releases or unlocks the resource which results into a deadlock because Process B is waiting for Process A to finish eventhough Resource X is not being used anymore.

When a deadlock happens, the best action that a system can probably take is first, terminate or kill the process where the deadlock occured. This solution however would result into possible loss of data or computations during execution because of the unfinished process. Another action that could be made by the system is to prevent deadlocks to occur by allocating more resources to monitor the processes that might get deadlocks.

1. In the producer-consumer example of Section 13.3, suppose that we incorrectly replaced the release(access) in the consumer process with wait(access). What would be the result of this error on execution of the system?

If the example of Section 13.3 used a 'wait(access)' line instead of 'release(access)', the execution will result into a deadlock because release(access) will let go of the resource while wait(access) will technically stay on its current event since it believes that the resource is not yet released.

1. Compare the Java mechanisms for cooperation and competition synchronization with that of Ada.

The mechanisms for cooperation and competition synchronization of Java and Ada are implemented differently although both support concurrent programming. The differences are:

* Thread support in Java is based of predefined classes while Ada has built in task features which is independent of object oriented programming model.
* Java makes use of 'monitor' and 'synchronized' methods while Ada uses MuTex or mutual exclusion through objects that are protected.
* Java makes use of the low level waiting methods while Ada uses condition based communication and synchronization through entries that are protected.
* Java and Ada use timing control but Ada supports custom scheduling policies.

To summarize, the concurrency model for Ada is more efficient for real-time applications compared to Java's.

4. What happens if a monitor procedure calls another procedure in the same monitor?

When a monitor procedure calls another procedure in the same monitor, it will be blocked and queued because a monitor is synchronized when it comes to accesses. This means that only one access is allowed as per execution.

References:

http://stackoverflow.com/questions/34512/what-is-a-deadlock

Concepts of programming languages: Robert W. Sebesta

<http://www0.cs.ucl.ac.uk/staff/w.emmerich/publications/ESEC01/ModelChecking/esec.pdf>

**Task C**

2. Design your solution using UML before you start programming:

a. What are your classes, instance variables, methods?

FishBuffer.java (Variables)

* occupied : int
* eWrite : int
* eRead : int
* Window : int[]

FishBuffer.java (Methods)

* get() : synchronized int
* set() : synchronized void
* Output() : void

CustomerProcessor.java (Variables)

* generator : Random
* buffer : FishBuffer
* numCustomers: int

CustomerProcessor.java (Methods)

* run() : void (Override from Thread)

MainExecutor.java (Methods)

* main() : void

b. What classes will extend threads?

CustomerProcessor.java

c. How do you synchronize the shared resource(s)?

Synchronization is implemented because of the concept called ‘monitors’. Each object has a monitor where threads can lock or unlock it so multiple accesses at the same time will not be allowed. If the object is locked during a thread execution, it will wait until the object will be unlocked before it can work with the shared resource.

In the programming side, by implementing the synchronized keyword in both get and set methods under the FishBuffer class, the process will be able continue in a stable manner.

d. What are the inputs from the user and/or from an initial configuration file? (well-designed code should not contain hard coded values)

* Number of customers to be processed.
* Number of employees.

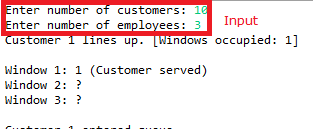
e. What values should be displayed to illustrate the behavior of your multi-threaded solution?

* What customer (in number) lined up?
* Status of employee windows
* Customers are waiting
* Employees are waiting

f. What type of interface (GUI, console)?

Console

g. What would sample dialogs look like?



Test Case 1: Customers: 10, Employees: 2

Customer 1 lines up. [Windows occupied: 1]

Window 1: 1 (Customer served)

Window 2: ?

Customer 1 entered queue.

Delaying for 11 ms.

Customer 2 lines up. [Windows occupied: 2]

Window 1: 1 (Customer going to be processed)

Window 2: 2

Customer 2 entered queue.

Delaying for 581 ms.

Serving customer 1 [Windows occupied: 1]

Window 1: 1 (Customer queued)

Window 2: 2 (Customer served)

Customer 1 left the queue

Customer 3 lines up. [Windows occupied: 2]

Window 1: 3

Window 2: 2 (Customer going to be processed)

Customer 3 entered queue.

Delaying for 131 ms.

Customers waiting...

Serving customer 2 [Windows occupied: 1]

Window 1: 3 (Customer served)

Window 2: 2 (Customer queued)

Customer 4 lines up. [Windows occupied: 2]

Window 1: 3 (Customer going to be processed)

Window 2: 4

Customer 4 entered queue.

Delaying for 930 ms.

Customer 2 left the queue

Serving customer 3 [Windows occupied: 1]

Window 1: 3 (Customer queued)

Window 2: 4 (Customer served)

Customer 3 left the queue

Serving customer 4 [Windows occupied: 0]

Window 1: 3 (Customer going to be processed)

Window 2: 4

Customer 4 left the queue

Customer 5 lines up. [Windows occupied: 1]

Window 1: 5 (Customer served)

Window 2: 4 (Customer queued)

Customer 5 entered queue.

Delaying for 766 ms.

Serving customer 5 [Windows occupied: 0]

Window 1: 5

Window 2: 4 (Customer going to be processed)

Customer 5 left the queue

Customer 6 lines up. [Windows occupied: 1]

Window 1: 5 (Customer queued)

Window 2: 6 (Customer served)

Customer 6 entered queue.

Delaying for 289 ms.

Customer 7 lines up. [Windows occupied: 2]

Window 1: 7

Window 2: 6 (Customer going to be processed)

Customer 7 entered queue.

Delaying for 652 ms.

Serving customer 6 [Windows occupied: 1]

Window 1: 7 (Customer served)

Window 2: 6 (Customer queued)

Customer 6 left the queue

Customer 8 lines up. [Windows occupied: 2]

Window 1: 7 (Customer going to be processed)

Window 2: 8

Customer 8 entered queue.

Delaying for 862 ms.

Serving customer 7 [Windows occupied: 1]

Window 1: 7 (Customer queued)

Window 2: 8 (Customer served)

Customer 7 left the queue

Customer 9 lines up. [Windows occupied: 2]

Window 1: 9

Window 2: 8 (Customer going to be processed)

Customer 9 entered queue.

Delaying for 318 ms.

Customers waiting...

Serving customer 8 [Windows occupied: 1]

Window 1: 9 (Customer served)

Window 2: 8 (Customer queued)

Customer 8 left the queue

Customer 10 lines up. [Windows occupied: 2]

Window 1: 9 (Customer going to be processed)

Window 2: 10

Customer 10 entered queue.

Delaying for 455 ms.

Serving customer 9 [Windows occupied: 1]

Window 1: 9 (Customer queued)

Window 2: 10 (Customer served)

Customer 9 left the queue

Serving customer 10 [Windows occupied: 0]

Window 1: 9 (Customer going to be processed)

Window 2: 10

Customer 10 left the queue

Total customers processed: 10

Test Case 2: Customers: 10, Employees: 3

Customer 1 lines up. [Windows occupied: 1]

Window 1: 1 (Customer served)

Window 2: ?

Window 3: ?

Customer 1 entered queue.

Delaying for 52 ms.

Customer 2 lines up. [Windows occupied: 2]

Window 1: 1 (Customer served)

Window 2: 2

Window 3: ?

Customer 2 entered queue.

Delaying for 636 ms.

Serving customer 1 [Windows occupied: 1]

Window 1: 1

Window 2: 2 (Customer served)

Window 3: ?

Customer 1 left the queue

Customer 3 lines up. [Windows occupied: 2]

Window 1: 1 (Customer queued)

Window 2: 2 (Customer served)

Window 3: 3

Customer 3 entered queue.

Delaying for 629 ms.

Serving customer 2 [Windows occupied: 1]

Window 1: 1 (Customer queued)

Window 2: 2

Window 3: 3 (Customer served)

Customer 2 left the queue

Serving customer 3 [Windows occupied: 0]

Window 1: 1 (Customer going to be processed)

Window 2: 2

Window 3: 3

Customer 3 left the queue

Customer 4 lines up. [Windows occupied: 1]

Window 1: 4 (Customer served)

Window 2: 2 (Customer queued)

Window 3: 3

Customer 4 entered queue.

Delaying for 360 ms.

Customer 5 lines up. [Windows occupied: 2]

Window 1: 4 (Customer served)

Window 2: 5

Window 3: 3 (Customer queued)

Customer 5 entered queue.

Delaying for 780 ms.

Serving customer 4 [Windows occupied: 1]

Window 1: 4

Window 2: 5 (Customer served)

Window 3: 3 (Customer queued)

Customer 4 left the queue

Serving customer 5 [Windows occupied: 0]

Window 1: 4

Window 2: 5

Window 3: 3 (Customer going to be processed)

Customer 5 left the queue

Customer 6 lines up. [Windows occupied: 1]

Window 1: 4 (Customer queued)

Window 2: 5

Window 3: 6 (Customer served)

Customer 6 entered queue.

Delaying for 791 ms.

Serving customer 6 [Windows occupied: 0]

Window 1: 4 (Customer going to be processed)

Window 2: 5

Window 3: 6

Customer 6 left the queue

Customer 7 lines up. [Windows occupied: 1]

Window 1: 7 (Customer served)

Window 2: 5 (Customer queued)

Window 3: 6

Customer 7 entered queue.

Delaying for 605 ms.

Customer 8 lines up. [Windows occupied: 2]

Window 1: 7 (Customer served)

Window 2: 8

Window 3: 6 (Customer queued)

Customer 8 entered queue.

Delaying for 193 ms.

Customer 9 lines up. [Windows occupied: 3]

Window 1: 7 (Customer going to be processed)

Window 2: 8

Window 3: 9

Customer 9 entered queue.

Delaying for 735 ms.

Serving customer 7 [Windows occupied: 2]

Window 1: 7 (Customer queued)

Window 2: 8 (Customer served)

Window 3: 9

Customer 7 left the queue

Customer 10 lines up. [Windows occupied: 3]

Window 1: 10

Window 2: 8 (Customer going to be processed)

Window 3: 9

Customer 10 entered queue.

Delaying for 175 ms.

Serving customer 8 [Windows occupied: 2]

Window 1: 10

Window 2: 8 (Customer queued)

Window 3: 9 (Customer served)

Customer 8 left the queue

Serving customer 9 [Windows occupied: 1]

Window 1: 10 (Customer served)

Window 2: 8 (Customer queued)

Window 3: 9

Customer 9 left the queue

Serving customer 10 [Windows occupied: 0]

Window 1: 10

Window 2: 8 (Customer going to be processed)

Window 3: 9

Customer 10 left the queue

Total customers processed: 10