Instituto Tecnológico Superior de Jerez

Jerez de García Salinas Zac.

07 de junio de 2021

Clara Ruby Pinedo Berumen

S19070080

clararubipb15@gmail.com

Ingeniería en Sistemas Computacionales

Tópicos avanzados de programación

4° Semestre

Actividad 1 – Mapa Conceptual

M.T.I Salvador Acevedo Sandoval

Exercises

1. Fill in the blanks in each of the following statements:

a) A thread enters the *terminated* state when _____ when it successfully completes its task or otherwise terminates (perhaps due to an error)._____.

b) To pause for a designated number of milliseconds and resume execution, a thread should call method ____sleep_____ of class ____Thread_____.

c) Method _____signal_____ of class Condition moves a single thread in an object's *waiting* state to the *runnable* state.

d) Method ____signalAll_____ of class Condition moves every thread in an object's *waiting* state to the *runnable* state.

e) A(n) _____runnable_____ thread enters the _____terminated_____ state when it completes its task or otherwise terminates.

f) A *runnable* thread can enter the _____time waiting_____ state for a specified interval of time.

g) At the operating-system level, the *runnable* state actually encompasses two separate states, _____ready_____ and _____running_____ .

h) Runnables are executed using a class that implements the _____ExecutorService_____ interface.

i) ExecutorService method _____shutdown_____ ends each thread in an ExecutorService as soon as it finishes executing its current Runnable, if any.

j) A thread can call method _____await_____ on a Condition object to release the associated Lock and place that thread in the _____waiting_____ state.

k) In a(n) _____producer/consumer_____ relationship, the _____productor_____ generates data and stores it in a shared object, and the ____consumer_____ reads data from the shared object.

l) Class _____ArrayBlockingQueue_____ implements the BlockingQueue interface using an array.

m) Keyword _____Synchronize_____ indicates that only one thread at a time should execute on an object.

2. State whether each of the following is *true* or *false*. If *false*, explain why.

a) A thread is not *runnable* if it has terminated. true

b) Some operating systems use timeslicing with threads. Therefore, they can enable threads to preempt threads of the same priority. true

c) When the thread's quantum expires, the thread returns to the *running* state as the operating system assigns it to a processor. true

d) On a single-processor system without timeslicing, each thread in a set of equal-priority threads (with no other threads present) runs to completion before other threads of equal priority get a chance to execute. true

3. *(True or False)* State whether each of the following is *true* or *false*. If *false*, explain why.

a) Method sleep does not consume processor time while a thread sleeps. true

b) Declaring a method synchronized guarantees that deadlock cannot occur. true

c) Once a ReentrantLock has been obtained by a thread, the ReentrantLock object will not allow another thread to obtain the lock until the first thread releases it. true

d) Swing components are thread safe. false, due to they cannot be manipulated by multiple without the risk of incorrect results.

4. *(Multithreading Terms)* Define each of the following terms.

a) thread A thread is said to be in one of several thread states.

b) multithreading Multiple threads of execution, where each thread has its own method-call stack and program counter, allowing it to execute concurrently with other threads while sharing with them application-wide resources such as memory.

c) *runnable* state A thread in the runnable state is considered to be executing its task.

d) *timed waiting* state A runnable thread can enter the timed waiting state for a specified interval of time. It transitions back to the runnable state when that time interval expires or when the event it's waiting for occurs.

e) preemptive scheduling When a higher-priority thread enters the ready state, the operating system generally preempts the currently running thread.

f) Runnable interface interface specify a task that can execute concurrently with other tasks.

g) notifyAll method To notify waiting threads to become runnable.

h) producer/consumer relationship a producer thread generates data and places it in a shared object called a buffer. A consumer thread reads data from the buffer.

i) quantum Time from running to ready state.

5. *(Multithreading Terms)* Define each of the following terms in the context of Java's threading mechanisms:

a) synchronized there's no way to explicitly state the condition on which threads are waiting, and thus there's no way to notify threads waiting on one condition that they may proceed without also signaling threads waiting on any other conditions.

b) producer A producer thread generates data and places it in a shared object called a buffer.

c) consumer A consumer thread reads data from the buffer.

d) wait This releases the monitor lock on the object, and the thread waits in the waiting state while the other threads try to enter the object's synchronized statement(s) or method(s).

e) notify To allow a waiting thread to transition to the runnable state again.

f) Lock Allow you to interrupt waiting threads or to specify a timeout for waiting to acquire a lock, which is not possible using the synchronized keyword.

g) Condition Allow you to specify multiple conditions on which threads may wait.

Concurrencia

¿Qúe es?

Es la distribución de procesos y el orden sin que se afecte el resultado final.

ExecutorService

Los **hilos** se utilizan mediante el executor service y tienen 2 estados, **ejecutandose** y **terminado.**

Maneja

Productor.

Crea datos y los almacena en un objeto.

Consumidor.

Lee los datos del objeto compartido.

ArrayBlockingQueue

Implementa.

Lock

Interrumpe los hilos y especifica un tiempo de espera.

Condition

Pone condiciones para las ejecuciones de los hilos.

Métodos.

Wait

Quita el bloqueo del monitor del objeto.

Notify

Permite a un subproceso volver a ser ejecutable.

NotifyAll

Permite que los subprocesos sean ejecutables.

Métodos

Sleep.

Deja el hilo en reposo.

Signal.

Ejecuta un hilo.

SignalAll.

Ejecuta todos los núcleos que están en reposo.