PREGUNTAS DE EXAMENES DE PROGRAMAICÓN

PARTE TEORICA:

(1,5 puntos) Explicar el patrón de diseño Future y su implementación en Java.

(2 puntos) Mostrar el uso de locks explícitos mediante un ejemplo de implementación de una Región Crítica.

(2 puntos) Enumerar y comparar los tipos de pool de hilos existentes en Java.

(1 punto) Explicar las diferencias entre Runnable y Callable. (Brevemente)

(3 puntos) Mostrar el uso de semáforos Java mediante un ejemplo de sincronización de dos procesos. Un proceso A debe esperar a que un proceso B le indique que ya puede continuar. Incluir todos los comentarios y código de relleno para que el ejemplo sea comprensible.

(2 puntos) Plantear dos posibles políticas de ejecución válidas para un pool de Java pero que no estén incluidas en los pools generados por los métodos de factoría de Executors.

(3 puntos) Explicar las diferencias entre monitor de Hoare y monitor simple (o de Mesa). Mostrar el código de Java para un monitor simple que presenta el problema que hace más seguros los monitores de Hoare.

(1 punto) Explicar brevemente las distintas versiones de Colas concurrentes basadas en la interfaz BlockingQueue.

(1 punto) Comparar e indicar las diferencias entre crear un HashMap mediante Collections.synchronizedMap y utilizando la clase ConcurrentHashMap (utilizar como máximo una cara).

(1 punto) Definir qué es una situación de interbloqueo y las 4 condiciones necesarias para que se produzca. Indicar posibles consecuencias de esta situación y posibles soluciones (utilizar como máximo una cara).

(1 punto) Explicar las diferencias entre la ejecución de tareas en pools de hilos o mediante la creación directa de hilos (utilizar como máximo una cara).

PARTE PRACTICA:

(3 puntos) Realizar un programa Java completo que implemente un productor-comsumidor con 1 productor y 3 consumidores, donde el productor produce 1 dato por ciclo y lo envía, el mismo dato, a cada uno de los consumidores. Usaremos 3 medios distintos para comunicar el productor con cada uno de los tres consumidores:

- 1 colección no concurrente pero segura
- 1 cola concurrente
- 1 colección concurrente

Requisitos generales a cumplir:

- El único elemento static del programa será el método main
- El programa deberá funcionar a excepción de posibles errores menores "de compilación"
- Se incluirá un diagrama de clases con los nombres, un diagrama de organización que represente el funcionamiento del programa y comentarios en el código suficientes para entender su funcionamiento.
- Para el resto de las decisiones el alumno aplicará sus conocimientos y sentido común, indicando lo decidido y justificándolo

---

(3 puntos) Una Agencia de servicios turísticos funciona de la siguiente manera:

- Dispone de 10 minibuses con capacidad para 8 viajeros.
- Cuando un viajero quiere visitar Alcalá, se pone a la cola para montar en un minibús.
- A la puerta de la agencia se pone un minibús libre que espera hasta que se completa con 8 viajeros. Entonces realiza su recorrido de 10-15 segundos y a continuación se bajan los viajeros, quedando libre para otro viaje.
- En cuanto parte uno, se pone otro libre a la puerta a esperar nuevos viajeros. Si los diez minibuses están ocupados, los viajeros esperan en la cola.
- Cuando un viajero desciende de un minibús, da un paseo de 30 a 80 segundos y vuelve a la puerta de la agencia para otro viaje.

Los viajeros son threads de la clase Viajero que llegan en momentos al azar a la Agencia y ejecutan el método verAlcala().

Se pide escribir la clase Agencia con dicho método y todo lo necesario para seguir el funcionamiento descrito.

También se pide escribir la clase Viajero y un programa Principal que cree 50 viajeros y los active de uno en uno cada 5-10 segundos.

Por último, explicar cómo funciona el sistema y las herramientas de sincronización usadas.

(3 puntos) Se pretende simular un cruce de calles por el que circulan coches de oeste a este y de norte a sur. Para regular el tráfico del cruce hay dos semáforos, uno en la entrada oeste y otro en la entrada norte, y dos sensores que se activan cuando llega un coche a la entrada correspondiente. También hay sensores que indican la salida del cruce. Se desea desarrollar un monitor en Java que simule la gestión del cruce de la siguiente forma:

- Los coches deben llegar al semáforo desde el norte (los que van hacia el sur) o desde el oeste (los que van hacia el este) de forma aleatoria.
- Si el semáforo correspondiente está en verde, el coche pasa inmediatamente. Si está en rojo, espera hasta que esté verde.
- Los coches tardan un cierto tiempo en pasar el cruce (un tiempo aleatorio entre 2 y 4 segundos). Una vez que el semáforo se pone en verde los coches pueden cruzar, pero siempre y cuando no haya un coche terminando de atravesar el cruce en sentido transversal.
- Sólo uno de los dos semáforos (oeste o norte) puede estar en verde a la vez para evitar colisiones de los coches.
- Cada 10 segundos ambos semáforos cambian su color de forma automática (de rojo a verde, y de verde a rojo).
- Para simular el comportamiento se deben crear 40 coches que llegan al cruce

---

(3 puntos) Una compañía de producción de prendas de vestir desea simular el proceso de doblado y empaquetado de prendas. En este sistema se distinguen dos tipos de tareas: doblar cada prenda que llega y empaquetar prendas cuando 5 del mismo tipo estén dobladas.

Se consideran 5 tipos de prendas: camisa, pantalón, camiseta, falda y chaqueta. El sistema dispone de 5 trabajadores para la tarea de doblado, pero si no hay tareas, hasta 4 de ellos podrían pasar a otros trabajos, quedando solo 1 a la espera. Para la tarea de empaquetado se dispone de 3 trabajadores que si no hay tareas bajará hasta 1 que queda a la espera.

El sistema no necesita llevar un control individual de cada prenda, sólo conocer el número de las que están pendientes o de las que se han doblado de cada tipo o la cantidad de paquetes de cada tipo que hay en el sistema. Crear el sistema con 200 prendas de prueba que "llegan" con una frecuencia aleatoria.

(3 puntos) Se desea modelar el comportamiento de un grupo de jugadores de fútbol que tienen un partido concertado y los árbitros que, además, son los responsables de las instalaciones. Se cuenta con 22 jugadores que van llegando de forma escalonada (tiempo aleatorio entre 5-20s) y los 5 árbitros que, desde el principio están en las instalaciones, esperando a que todos lleguen. Una vez reunidos los 22 jugadores deben esperar a que los 5 árbitros se preparen (tardan de 2s a 7s).

A continuación, comienza el partido, donde los 22 jugadores se ponen en marcha (transcurren 90s). Los árbitros esperan a que todos los jugadores terminen y los 5 árbitros finalizan con un mensaje indicando el final del partido. Se mostrará en pantalla cada cambio de estado, incluyendo la finalización de cada estado y el comienzo del siguiente.

(3 puntos) Se desea modelar el comportamiento de un conjunto de coches que van a hacer una carrera y los coches de seguridad (safety-car) que, además, son los responsables de la seguridad en la carrera.

Se cuenta con 20 coches de carreras y 2 coches de seguridad. Todos ellos van llegando al circuito de forma escalonada y en cualquier orden (tiempo aleatorio entre 4-16s). Una vez reunidos los 22 coches en el circuito se pasa a la fase de revisión del circuito. Los 2 coches de seguridad inician una vuelta de revisión del circuito para comprobar que todo está en orden (tardan de 20s a 30s) mientras los 20 coches de carreras esperan a que terminen.

A continuación, comienzan la carrera los 20 coches de carreras (duración 120s). Los 2 coches de seguridad esperan a que todos los corredores terminen y los 2 coches de seguridad finalizan entonces con un mensaje indicando el final de la carrera.

Se mostrará en pantalla cada cambio de estado, incluyendo la finalización de cada estado y el comienzo del siguiente.

El programa se deberá modelar **sin utilizar monitores, semáforos ni cerrojos.**