Parcialito: concurrencia y sincronización

Modele utilizando thread el siguiente problema. Recuerde que debe:

1. Maximizar la concurrencia.
2. Evitar deadlocks.
3. Evitar race conditions.
4. Evitar busy-waits.

Debe entregar el código fuente que modele el comportamiento comprimido en un archivo zip. El archivo debe contener el código en las carpetas correspondiente a los paquetes.

El parque "La Piedra Oscilante" posee un museo y un parque con numerosas réplicas de la legendaria Piedra Oscilante en el que se hacen paseos en coches. Hay m visitantes al parque que desean hacer el paseo y n coches para realizarlo, cada uno de ellos con capacidad para 4 pasajeros. Los pasajeros deambulan durante un rato por el museo y luego hacen cola en la parada de los coches (hay una sola parada de la que salen todos los coches). Cuando un coche está disponible carga a sus 4 pasajeros y circula por el parque una cantidad de tiempo aleatoria. Durante el paseo, el coche puede fallar (con una probabilidad baja) en cuyo caso le informa al técnico para que vaya a reparar para poder continuar el paseo.

Considere que:

- Los visitantes se suben a los autos en el orden en que se agregaron a la cola.
- Si todos los coches están en uso, los pasajeros que desean usarlos esperan en la cola hasta que haya un coche disponible.
- Si el coche está disponible, pero no hay pasajeros en espera, el coche espera hasta que alguien desee subir.
- Si un coche está disponible y sólo hay uno, dos o tres pasajeros dispuestos a subir, todos deben esperar hasta completar los 4 pasajeros.
- El técnico descansa cuando no hay coches que reparar.
- Si 2 o más coches fallaron al mismo tiempo, el técnico reparará los coches en el orden que estos reportaron el problema.
- Hay un solo técnico.
- Se pueden utilizar semáforos, mutex (locks), y monitores. No pueden utilizarse estructuras thread-safe provistas por Java, como aquellas que heredan de BlockingQueue.