CC60H : Nuevos Lenguajes de Programación
Control 2–Semestre Primavera'99
Prof.: Luis Mateu.

Sin apuntes, 1 hora 20 minutos

Pregunta 1

Un programador ha escrito la siguiente versión de un Lock:

```
public class Lock {
  int turno= 0;
  int atendidos= 0;
  public void lock() {
    int miTurno= turno;
    turno++;
    while (atendidos!= miTurno)
      ;
  }
  public void unlock() {
    atendidos++;
  }
}
```

Por medio de un diagrama de threads muestre que este programa es incorrecto. Luego corrija el programa introduciendo el mínimo de modificaciones.

Pregunta 2

Escriba un programa que cree un objeto de la clase Lock y lo publique de modo que pueda ser utilizado en forma remota. De ser necesario, señale qué cambios se deben hacer a su respuesta de la pregunta anterior. Haga un bosquejo que indique la manera en que un proceso de otra máquina podría utilizar ese objeto.

Preócupese de los aspectos esenciales de la programación, ignorando detalles como los import, las excepciones, los *security managers* o los comandos que lanzan los programas. (En otras palabras, se le está preguntando cuáles son esos aspectos que sí son esenciales.)

Pregunta 3

Una de las implementaciones de un servidor IRC que se estudió en clases consiste en lo siguiente: los clientes se subscriben al servidor suministrando un objeto *listener*. El servidor se queda con una referencia de este objeto que reside remotamente en el cliente. Cada vez que el servidor recibe un nuevo mensaje de parte de algún cliente, el servidor invoca el método `msgArrived` de todos los objetos *listener* que estén registrados en el servidor.

Se estudiaron dos variantes. La primera ejecutaba secuencialmente (en un solo thread) el método `msgArrived` de cada *listener*. La segunda variante creaba un thread por cada cliente, en donde invocaba concurrentemente el método `msgArrived` de cada *listener*.

El mismo programador de la pregunta 1 argumenta que la segunda variante es inútil en un monoprocesador, porque finalmente será el mismo procesador el que deberá ejecutar todas las invocaciones remotas. Incluso sostiene que la creación de tantos threads agrega un sobrecosto que hace que finalmente esa variante sea más lenta que la que envía todos los mensajes secuencialmente.

Discuta si el programador tiene razón o no. Si Ud. cree que el programador está equivocado, explique claramente en donde se puede ganar velocidad en la segunda variante.

Pregunta 4

Indique en el siguiente programa qué `catch` atrapará cada una de las instrucciones que lanzan excepciones:

```
     ... void main(...) {   |      void Q() throws Exception {
       try {                |        try {
         ...                |          ...
(1)      throw new ExcpB(); |  (2)     throw new ExcpA();
         Q();               |          ...
         ...                |        }
       }                    |  (C)   catch (ExcpB e) {
(A)    catch (ExcpA e) {    |          ...
         ...                |        }
       }                    |      }
(B)    catch (ExcpB e) {    |
         ...                |
       }                    |
     }                      |
```

Analice el código anterior cuando:

1. ExcpA no es subclase de ExcpB y ExcpB no es subclase de ExcpA.
2. ExcpA es subclase de ExcpB.
3. ExcpB es subclase de ExcpA.

Ayuda memoria:

```
public class Naming {
  public static Remote lookup(String url);
  public static void rebind(String url, Remote obj);
}

public interface Remote { ... }
```

```
public class UnicastRemoteObject { ... }

public class Object {
  public void wait();
  public void notify();
  public void notifyAll();
}
```