Universidad Rey Juan Carlos

Programación Concurrente
3º Grado en Ingeniería de Computadores

30 de Junio 2014

Apellidos:	
Nombre:	**DNI:**

La duración máxima del examen será de 2:00 horas.

Examen 2

Ejercicio 1) Concurrencia en JavaScript (2p)

Describe qué mecanismos existen en JavaScript para programar de forma concurrente. ¿Estos mecanismos están disponibles en los navegadores y en plataformas como node.js?

Ejercicio 2) Estructuras de datos en Java (3p)

2.a) ¿Cualquier estructura de datos del Java Collections Framework se puede compartir entre varios hilos o tiene que cumplir algún requisito especial? (1p)

2.b) Supongamos que hemos desarrollado un programa en Java en el que un mapa está compartido por varios hilos. Ahora necesitamos ampliar el programa para que los hilos puedan insertar una clave en el mapa (key) sólo si esa clave no se encuentra ya incluida en él. ¿Cómo se podría implementar ese fragmento de código? Si existen varias formas de usar un mapa compartido en Java, explica cómo se implementaría en cada una de ellas (3p).

Ejercicio 3) Programa concurrente (4p)

Se desea implementar una clase en Java para controlar los turnos en un juego concurrente. En ese juego, cada jugador está representado como un hilo de ejecución. La clase tiene el siguiente interfaz:

```
public class Turno {

    //El método se bloquea hasta que sea el turno del jugador o se acabe la partida
    public boolean esperaTurnoOFin(int numJugador){ … }

    //Este método será invocado por un jugador para indicar que se acaba la partida
    public void finPartida(){ … }

    //Este método será invocado por un jugador para pasar el turno al siguiente jugador
    public void siguienteTurno(){ … }
}
```

Cada uno de los jugadores llamará al método esperaTurnoOFin(). Ese método quedará bloqueado hasta que sea el turno del jugador indicado como parámetro (devolviendo false) o bien el juego ha finalizado (devolviendo true). El jugador con el turno podrá invocar el método siguienteTurno() o el método finPartida().

Un jugador podría usar esta clase de la siguiente forma:

```
Turno turno = …
while(true){
    boolean fin = turno.esperaTurnoOFin(numJugador);
    if(fin){
        break;
    } else {
        //Juego...
        if(...){
           turno.finPartida();
           break;
        } else {
           turno.siguienteTurno();
        }
    }
}
```

Se pide implementar completamente la clase Turno para que se comporte como pide el enunciado.

Solución: