SISTEMI OPERATIVI IIN/IEL/IDT

INFORMATICA INDUSTRIALE E SISTEMI OPERATIVI IDI

SISTEMI DI ELABORAZIONE P.O.

prova scritta del 18.02.2005

Nome: _____ Cognome: _____

Si realizzi un programma Java che all'avvio crea un'istanza della classe *Tavolo*; durante la sua inizializzazione l'oggetto di tipo *Tavolo* sceglie a caso un “numero segreto”, un intero compreso tra 0 e M. Una volta creato il tavolo, il programma provvede ad avviare N thread, tutti istanze della classe *Indovino*, il cui comportamento è descritto qui di seguito.

All'inizio della propria esecuzione ciascun thread provvede a comunicare al Tavolo che è stato avviato e quindi si sospende in attesa che siano avviati anche gli altri thread. Solo quando tutti gli N thread saranno stati avviati questi possono procedere a svolgere la loro attività, ovvero indovinare il numero inizialmente scelto dal tavolo. In particolare, la logica di ciascun thread prevede che questo verifichi anzitutto con il tavolo se il numero è già stato indovinato (a questo scopo si suggerisce di definire un opportuno metodo *indovinato(...)* per la classe *Tavolo*); quindi, in caso di risposta positiva termina la propria esecuzione, mentre in caso di risposta negativa sceglie un numero a caso e chiede al tavolo di confrontarlo con il numero segreto (a questo scopo si suggerisce di definire un opportuno metodo *confronta(...)* per la classe *Tavolo*); nel caso in cui il numero coincide, il tavolo registra il fatto che il numero è stato indovinato, mentre il thread termina la propria esecuzione; qualora invece il thread non avesse indovinato il numero, esso cederà il controllo ad un altro thread, e quindi attenderà che si presenti nuovamente il proprio turno.

Nella definizione dei metodi e nell'implementazione degli stessi si raccomanda di concentrarsi sulle problematiche di sincronizzazione.

Soluzione

```
/**
 * Tavolo di gioco.
 * Il tavolo tiene traccia degli indovini che partecipano al gioco, e consente l'avvio del gioco solo
 * una volta che tutti gli indovini previsti si sono presentati al tavolo.
 * Il tavolo confronta anche con il numero segreto i numeri presentati da ciascun indovino durante il
 * proprio turno
 */
class Tavolo
{
	// numero segreto da indovinare
	private int numeroSegreto;
	// indica se il numero segreto e' stato indovinato o meno
	private boolean indovinato = false;
	// numero di indovini che si devono presentare al tavolo
	private int indovini;

	public Tavolo  (int nm, int i) {
		numeroSegreto = (int) ( nm * Math.random() );
		indovini = i;
		System.out.println( "Il numero segreto e' " + numeroSegreto );
	}

	/**
	 * invocando il metodo gli indovini comunicano al tavolo che sono stati avviati
	 */
	public synchronized void comunicaAvvio() {
		// decrementa il numero di indovini che si devono ancora presentare
		indovini--;

		// se qualche indovino si deve ancora presentare...
		if ( 0 != indovini ) {
			// ...sospende l'indovino...
			try {
				this.wait();
			}
			catch ( InterruptedException ie ) {
			}
		}
		// ...altrimenti sblocca tutti gli indovini sospesi
		else
			notifyAll();
	}

	/**
	 * confronta con il numero segreto il numero indicato dall'indovino.
	 *
	 * @param i il numero indicato dall'indovino
	 * @return true se l'indovino ha indovinato il numero, false altrimenti
	 */
	public synchronized boolean confronta( int i ) {
		if ( i == numeroSegreto )
			indovinato = true;
		return indovinato;
	}
```

```
	/**
	 * indica se il numero segreto e' già stato indovinato da qualche altro indovino.
	 *
	 * @return true se un indovino ha indovinato il numero, false altrimenti
	 */
	public boolean indovinato() {
		return indovinato;
	}
}


class Indovino extends Thread
{
	// riferimento al tavolo presso cui si svolge il gioco
	private Tavolo tavolo = null;
	// indica se il thread deve terminare
	private boolean finito = false;
	// estremo superiore dell'intervallo che comprende i numeri da indovinare [0,numerMax]
	private int numeroMax;

	public Indovino ( int nm, Tavolo t ) {
		tavolo = t;
		numeroMax = nm;
	}

	public void run() {
		// sincronizzazione iniziale (attende il "via" dal programma principale
		tavolo.comunicaAvvio();

		// finche' il numero non e' stato indovinato il gioco va avanti
		while ( ! finito ) {
			/* accesso sincronizzato al tavolo, per consentire l'invocazione dei
			   due metodi senza che fra le due invocazioni cambi lo stato del tavolo.
			   Inoltre, si sfrutta l'accodamento implicito nel monitor per gestire i turni
			 */
			synchronized( tavolo ) {
				if ( tavolo.indovinato() ) {
					System.out.println( "numero gia' indovinato");
					finito = true;
				}
				else {
					int n = (int) ( numeroMax * Math.random() );
					System.out.print( "Provo con " + n + "... " );
					if ( tavolo.confronta( n ) ) {
						System.out.println( "ho indovinato il numero !!!" + n);
						finito = true;
					}
					else
						System.out.println( "non ho indovinato il numero" );
				}
			}
		}
	}

}
```

```
/**
 * Programma principale.
 * Provvede alla creazione dei tavolo, crea ed avvia i thread degli indovini
 */
public class Programma
{
	// il numero di indovini
	public static final int N = 5;
	// limite superiore per l'intervallo dei numeri da indovinare [0,M]
	public static final int M = 20;

	public static void main( String[] args ) {
		// crea il tavolo
		Tavolo t = new Tavolo( M, N );
		// crea
		for ( int k = 0; k < N; k++ ) {
			Indovino i = new Indovino( M, t );
			i.start();
		}
	}
}
```