

CoDeK

Concurrent Development framework

<http://www.primianotucci.com/go/codek>

Primiano Tucci

The screenshot displays the CoDeK Concurrent Development framework interface. The window title is "Concurrent Development framework - Primiano Tucci - http://www.primianotucci.com". The interface includes a "Play/Step" button and a checked "step by step" option.

Threads: Six threads are shown in a row:

- 1 scende NEW (green border)
- 2 scende NEW (green border)
- 3 sale NEW (green border)
- 4 sale WAITING (orange border, Cond 1 Salita [Lock0])
- 5 scende NEW (green border)
- 6 scende WAITING (orange border, Cond 2 Discesa [Lock0])

Locks / Conditions: Five panels are shown:

- Lock0: UNLocked, Waiting Threads: none.
- Cond 1 Salita [Lock0]: Waiting Threads: 89 sale, 93 sale, 97 sale, 98 sale, 99 sale.
- Cond 1 Discesa [Lock0]: Waiting Threads: none.
- Cond 2 Salita [Lock0]: Waiting Threads: none.
- Cond 2 Discesa [Lock0]: Waiting Threads: 92 scende, 94 scende, 95 scende, 96 scende, 100 scende.

Data: Five data structures are shown:

- Sez1: Size: 1, 3 sale.
- Sez2: Size: 3, 1 scende, 5 scende, 2 scende.
- Piazz: Size: 0.
- CodaDisc 1: Size: 0.
- CodaDisc 2: Size: 44, 92 scende, 94 scende, 95 scende, 96 scende, 100 scende.

Console: Log output showing thread actions and state changes:

```
[1 scende] Sez1: 0D ; Sez2: 0D ; Piazz: 0 ; CodaDisc1: 0 ; CodaDisc2: 0
1 ha iniziato la discesa 2
[3 sale] Sez1: 0D ; Sez2: 1D ; Piazz: 0 ; CodaDisc1: 0 ; CodaDisc2: 0
3 ha iniziato la salita 1
[5 scende] Sez1: 1S ; Sez2: 1D ; Piazz: 0 ; CodaDisc1: 0 ; CodaDisc2: 0
5 ha iniziato la discesa 2
[2 scende] Sez1: 1D ; Sez2: 2D ; Piazz: 0 ; CodaDisc1: 0 ; CodaDisc2: 0
2 ha iniziato la discesa 2
```

Istruzioni di installazione / utilizzo

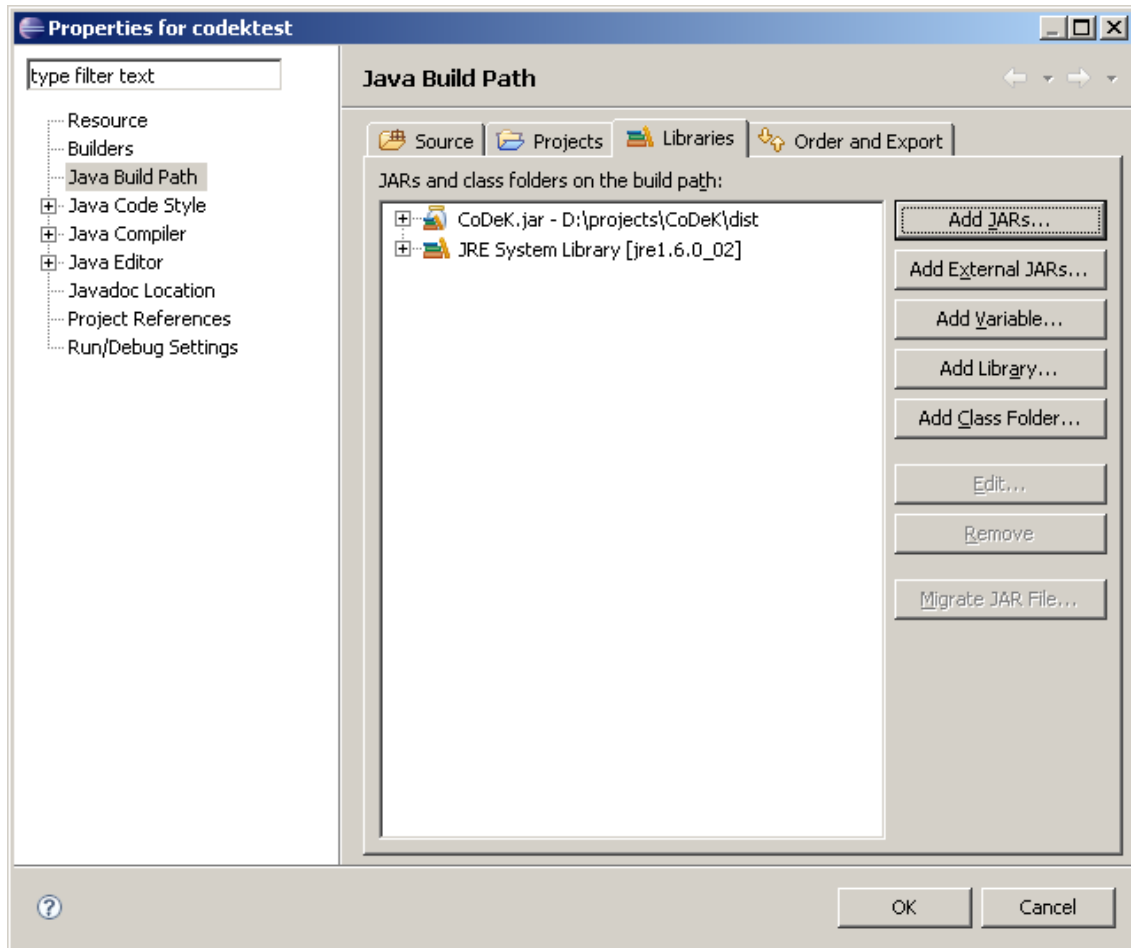
1. Importazione della libreria

Importare il file CoDeK.jar tra le librerie del progetto (assicurarsi di utilizzare almeno JDK 1.5).

In Eclipse:

Menu Project -> Properties -> Java Build Path

Pannello "Libraries" -> Add external JARs -> [aggiungere CoDeK.jar]



2. Utilizzo della libreria

Il principio di funzionamento della libreria di basa sul presupposto di subclassare alcune classi standard, quali Thread, ReentrantLock, Condition e Vector<> ed aggiungere automaticamente un monitor grafico che consente di monitorare lo stato dei Thread, verificare su quali variabili condizione essi sono eventualmente bloccati, ottenere la lista di Thread bloccati su una variabile condizione, ecc.

Il codice sorgente rimane assolutamente inalterato, è richiesta tuttavia una piccola modifica agli import nelle intestazioni dei file java.

In particolare è necessario sostituire i vari

```
import java.util.concurrent.ecc ecc
```

Con le seguenti due righe

```
import com.primianotucci.codek.*;  
import com.primianotucci.codek.Thread;
```

All'avvio del progetto, l'interfaccia grafica verrà automaticamente caricata e consentirà di monitorare l'applicazione.

Modalità step by step

In particolare è disponibile una modalità di esecuzione "step by step". Supponendo di aver utilizzato delle Thread.sleep(xx) per simulare nei vari punti del programma una fase di elaborazione, quando la modalità step by step è attiva, le procedure di sleep vengono automaticamente modificate e i Thread in sleep non proseguono fino a quando non viene (ri)cliccato il pulsante "Play/Step" sull'interfaccia grafica. Questo consentirà di analizzare "pezzo per pezzo" il funzionamento del programma, aiutando con molta probabilità l'individuazione di condizioni di deadlock ecc.

3. Consigli ed Esempi

Al fine di migliorare l'analisi dell'esecuzione, vengono riportati alcuni consigli da applicare durante il design (validi comunque a prescindere dall'utilizzo o meno della libreria).

- Non utilizzare mai il percorso completo del package in fase di creazione/istanziamento delle classi, ma effettuare un import ed utilizzare solo il nome della classe.

Questa raccomandazione è d'obbligo per utilizzare la libreria

Esempio:

```
java.util.concurrent.locks.ReentrantLock l= new  
java.util.concurrent.locks....
```

NO

```
import java.util.concurrent.locks.*;
```

Oppure, nel caso di utilizzo della libreria

```
import com.primianotucci.codek.*  
import com.primianotucci.codek.Thread  
  
....  
ReentrantLock l=new ReentrantLock();
```

SI'

- Utilizzare una lista dinamica (Vector<x>) per tenere traccia degli elementi in coda / prodotti / consumati ecc.
In particolare, nel caso di utilizzo della libreria, tutte le istanze di Vector vengono monitorate ed il loro contenuto viene mostrato nell'interfaccia grafica.

Esempio:

```
Vector<Persona> persone_in_attesa = new Vector<Persona>();  
  
void inizia_salita(Persona p) {  
    ...  
    persone_in_attesa.add(p);  
    ...  
}  
  
void termina_salita(Persona p) {  
    ...  
    persone_in_attesa.remove(p);  
    ...  
}
```

- Assegnare dei nomi ai Thread creati.
Sebbene in Java esistano diverse modalità di creazione dei Thread, in ogni caso è possibile (a prescindere dall'utilizzo della libreria) e consigliabile assegnare dei nomi ai Thread.
Ciò può essere fatto sia in fase di creazione del Thread, passando la stringa del nome come secondo parametro, o successivamente, utilizzando il metodo setName("nome del thread").

Esempio:

```
Thread thdPoll;
...
<funzione nella quale si sta istanziando il thread>{

    thdPoll = new Thread( new Runnable(){
        public void run(){
            procedura_del_thread();
        }
    });
    thdPoll.setName("Buffer polling Thread");
    thdPoll.start();

}

private void procedura_del_thread(){
    while(...){
        ...
        ...
        ...
    }
}
```