Problematica Produttore-Consumatore con visualizzazione in **JTextArea**

```
/**
 * @(#)Ideal.java
 *
 * Ideal application
 *
 * gestione di eventi di azione: pulsante per resettare l'area di testo
 *
 * @author 4AI_SIS
 * @version 1.00 2011/3/25
 */
import java.awt.event.*;
import java.awt.*;
import javax.swing.*;

 // Classe Magazzino ****************************
// che si vuole vuoto

class WareHouse{
private int numberOfProducts;
private int idProduct;
private boolean empty = true; // magazzino vuoto
private JTextArea a;

public WareHouse (JTextArea ta){
          a= ta;
}
public synchronized void put(int idProduct) {
if (!empty) // se il magazzino non è vuoto...
        try {
          wait(); // fermati Producer
        }   catch (InterruptedException exc) { exc.printStackTrace();  }
this.idProduct = idProduct;
numberOfProducts++;
printSituation("Produced " + idProduct);          // scrive
empty = false;
notify(); // svegliati Consumer
}

public synchronized int get() {
if (empty) // se il magazzino è vuoto...
       try {
         wait(); // bloccati Consumer
       }  catch (InterruptedException exc) {   exc.printStackTrace();   }
numberOfProducts--;
printSituation("Consumed " + idProduct);          // scrive
empty = true; // il magazzino ora è vuoto
notify(); // svegliati Producer
return idProduct;
}

private synchronized void printSituation(String msg) {                              // scrive in JTextArea
  a.append (msg +"\n" + numberOfProducts  + " Product in Warehouse\n");
}
}
```

```
//classe Produttore *************************

class Producer implements Runnable {

private WareHouse wareHouse;

public Producer(WareHouse wareHouse) {
        this.wareHouse = wareHouse;
        new Thread(this, "Producer").start();
}
public void run() {
        for (int i = 1; i <= 10; i++) {
                wareHouse.put(i);
        }
}
}

//classe Consumatore *************************
class Consumer implements Runnable {

private WareHouse wareHouse;

public Consumer(WareHouse wareHouse) {
        this.wareHouse = wareHouse;
        new Thread(this, "Consumer").start();
}
public void run() {
        for (int i = 0; i < 10; i++) {
            i = wareHouse.get();
        }
   }
}

//classe del main *************************
 public class Ideal extends JFrame implements ActionListener {

        private JTextArea ta;
        private Container c;
        private JPanel p;
        private JButton b;

        public Ideal (){

                c = getContentPane();
                p = new JPanel();
                p.setPreferredSize(new Dimension(180, 35));
                b = new JButton("Reset");
                b.addActionListener (this);

                ta = new JTextArea("Traccia d'esecuzione dei threads\n");
                ta.setEditable (true);
                JScrollPane sp = new JScrollPane(ta);
                sp.setPreferredSize(new Dimension(220, 120));   // larghezza, altezza
                c.setLayout(new FlowLayout());                                  //Layout the content pane
                c.setPreferredSize(new Dimension(280, 180));
                p.add(b);
```

```
            c.add(p);
            c.add(sp);
            pack();
            setVisible(true);
            setDefaultCloseOperation (JFrame.EXIT_ON_CLOSE);
        }

        public void actionPerformed (ActionEvent e)  {   // unico metodo da ridefinire

            ta.setText("");    // resetta l'area di testo
         }

         public static void main(String args[]) {

            Ideal ogg = new Ideal();

            WareHouse wareHouse = new WareHouse(ogg.ta);
            new Producer(wareHouse);
            new Consumer(wareHouse);
          }
}
```