Volatile

```
public class MemoryConsistencyVolatileExample {

/*
(Comportamento senza Sincronizzazione)
In un sistema con memoria cache, ogni thread potrebbe memorizzare una copia della
variabile done nella propria cache locale. Ciò significa che il thread principale
imposta done a true. Tuttavia, il thread worker potrebbe continuare a vedere done
come false se sta leggendo la variabile dalla propria cache locale, piuttosto che
dalla memoria principale. Questo porta il thread worker a rimanere bloccato nel ciclo
while (!done) anche dopo che done è stato impostato a true nel thread principale.
*/

	// private boolean done = false;

/*
Per risolvere questo problema, si può usare la parola chiave volatile in Java.
Dichiarare una variabile come volatile garantisce che le letture e le scritture della
variabile avvengano direttamente dalla memoria principale, evitando che i thread
utilizzino copie cache della variabile.
*/

	private static volatile boolean done = false;

	public static void main(String[] args) {
		Thread worker = new Thread(new MyImplementedThread() {
			@Override
			public void run() {
				int i = 0;
				while(!done) {
					i++;
					System.out.println("in the loop, done = false");
				}
				System.out.println("out of the loop, done = true");
				System.out.println("Worker thread finisced, i = " + i);
			}
		});
		worker.start();

	  try {
	      Thread.sleep(1000);
	  } catch (InterruptedException e) {
	      Thread.currentThread().interrupt();
	  }

	  done = true;
	  System.out.println("Main thread set done to true");

	}
}
```