**Esercizio 2**

Nel codice originale la sincronizzazione e’ implementata direttamente sul metodo **occupaBagno** della classe **ServiziPubblici**. Questo tipo di sincronizzazione e’ poco efficiente visto che abbiamo a disposizione 10 bagni per uomini e 10 per donne e ogni bagno puo’ essere libero o occupato indipendentemente dagli altri.

La soluzione e’ rimuovere la sincronizzazione dalla classe ServiziPubblici e spostarla direttamente sulla singola istanza del bagno.

public boolean occupaBagno(final boolean uomo) {

Modifica a classe Bagno rendondala Thread Safe, 2 varianti:

* Con monitor pattern
* Con uso di Atomic e idioma del CAS

Le prestazioni delle 2 implementazioni sono paragonabili, non si riscontrano particolari benefici a usare l’ Atomic.

class Bagno {  
 // VERSIONE CON MONITOR PATTERN  
// private boolean occupato = false;  
// public synchronized boolean provaOccupare() {  
// if (occupato)  
// return false;  
// this.occupato = true;  
// return true;  
// }  
// public synchronized void libera() {  
// this.occupato = false;  
// }  
  
 // VERSIONE CON ATOMIC  
 private AtomicBoolean occupato = new AtomicBoolean(false);  
 public boolean provaOccupare() {  
 while(true) {  
 if (occupato.get())  
 return false;  
  
 if(occupato.compareAndSet(false, true))  
 return true;  
 }  
 }  
  
 public void libera() {  
 this.occupato.set(false);  
 }  
}