



Uso di PMD per l'analisi statica del codice e sistemazione violazioni

Violazioni pre review:

Element	# Violations	# Violations/K...	# Violations/M...	Project
▼  progettosemaforo	22	449.0	5.50	Progetto_Testa_...
▼  Semaforo.java	22	709.7	7.33	Progetto_Testa_...
▶ LocalVariableCouldBeFinal	2	64.5	0.67	Progetto_Testa_...
▶ CyclomaticComplexity	1	32.3	0.33	Progetto_Testa_...
▶ MethodArgumentCouldBeFinal	2	64.5	0.67	Progetto_Testa_...
▶ AvoidLiteralsInIfCondition	1	32.3	0.33	Progetto_Testa_...
▶ BeanMembersShouldSerialize	1	32.3	0.33	Progetto_Testa_...
▶ OnlyOneReturn	5	161.3	1.67	Progetto_Testa_...
▶ CommentRequired	4	129.0	1.33	Progetto_Testa_...
▶ CommentSize	5	161.3	1.67	Progetto_Testa_...
▶ AddEmptyString	1	32.3	0.33	Progetto_Testa_...

Codice pre review:

```
package progettosemaforo;
public class Semaforo {
    //stato indica il colore attuale dei due semafori
    private /*@ spec_public @*/ int[] stato;

    //@ public invariant (\exists int i; i>0 && i<stato.length; stato[i] == 2);

    //costruttore: inizializzo entrambi i semafori a rossi
    //@ ensures stato != null;
    //@ ensures (\forall int i; i>0 && i<stato.length; stato[i] == 2);
    public Semaforo() {
        stato = new int[2];
        stato[0] = 2;
        stato[1] = 2;
    }

    //funzione changecolor: dato un intero per il colore e uno per la selezione del semaforo
    //permette di cambiare il colore del semaforo selezionato sul colore scelto, solo se
    //e' nella sequenza verde, giallo, rosso e da rosso a verde solo se l'altro semaforo
    //e' a rosso.
    //@requires sem >= 0;
    //@requires sem <= 2;
    //@requires color >=0;
    //@requires color <=2;
    //@ensures \result == true || \result == false;
    //@ensures (\forall int i; i>0 && i<stato.length && i!=sem; stato[i] == \old(stato[i]));
    public boolean changecolor(int sem, int color) {
        if (sem >= 0 && sem <= 2 && color >= 0 && color <= 2) {
            // da verde passo a giallo
            if (color == 1 && stato[sem] == 0) {
                stato[sem] = 1;
                return true;
            }
            // da giallo passo a rosso
            if (color == 2 && stato[sem] == 1) {
```

```


        stato[sem] = 2;
        return true;
    }
    // da rosso passo a verde, ma per questo caso devo verificare anche l'altro
    // semaforo
    // se sem = 1 allora l'altro è il sem 0
    if (sem == 1) {
        if (color == 0 && stato[sem] == 2 && stato[0] == 2) {
            stato[sem] = 0;
            return true;
        }
    }
    else {
        if (color == 0 && stato[sem] == 2 && stato[1] == 2) {
            stato[sem] = 0;
            return true;
        }
    }
    return false;
}
else {
    return false;
}
}

//funzione toString: permette di restituire tramite due char il colore dei semafori al momento in cui
//viene chiamata
//@ also
//@ ensures (\forall int i; i>0 && i<stato.length; stato[i] == \old(stato[i]));
@Override
public String toString() {
    char sem1 = stato[0] == 0 ? 'V' : stato[0] == 1 ? 'G' : 'R';
    char sem2 = stato[1] == 0 ? 'V' : stato[1] == 1 ? 'G' : 'R';
    return "" + sem1 + sem2;
}
}

```

Usando come linee guida le violazioni, sono stati aggiunti dei commenti prima della dichiarazione di package e prima della dichiarazione della classe, andando a esplicitare la loro funzione. Tutti i commenti devono essere i commenti per javadoc (blu in eclipse). Analogamente sono stati rivisti tutti i commenti prima del costruttore e delle funzioni, lasciando inalterati quelli relativi a jml. Sono stati anche limitati in larghezza e lunghezza. Successivamente sono stati resi *final* i parametri passati all'interno di *changeColor* e i due *char* in *toString*. Nel metodo *toString* è anche stato modificato il ritorno, dove anziché usare le doppie virgolette si è utilizzato *Character.toString(sem)*. Successivamente nel metodo *changeColor* si è passati ad avere solo un punto di return del valore, introducendo la variabile “cambiato” e poiché questo aggiungeva degli errori di dataflow (la variabile era appena inizializzata e veniva subito modificata), è stata utilizzata anche nelle condizioni degli if. Come ultima cosa è stata resa *transient* la variabile *stato*, che indica che in fase di serializzazione la variabile è inizialmente forzata ad essere null.

Violazioni post review:

Element	# Violations	# Violations/K...	# Violations/M...	Project
▼  progettosemaforo	2	40.8	0.50	Progetto_Testa_...
▼  Semaforo.java	2	64.5	0.67	Progetto_Testa_...
▶ CyclomaticComplexity	1	32.3	0.33	Progetto_Testa_...
▶ CommentSize	1	32.3	0.33	Progetto_Testa_...

Le violazioni che sussistono sono relative alla complessità del metodo *change_color* poiché vi sono molti if al suo interno e il programma segnala questa presenza elevata. Purtroppo a causa della natura dei controlli per eseguire un cambio di colore questi if non possono essere uniti o eliminati o trasformati in funzioni.

Il secondo errore è relativo ad una riga di una post condizione di jml che il programma riconosce come commento e chiede di poterlo mettere su più righe, cosa che non è possibile poiché si tratta di una condizione \forall e quindi indivisibile.