Testing e verifica del software – 27 Gennaio 2015

Magazzino

Un magazzino contiene **cinque** tipi di prodotti. Il magazzino può contenere al massimo **100** unità di ogni prodotto. Si possono aggiungere quantità di un certo prodotto ma al massimo **10** alla volta e solo fino a raggiungere il livello massimo.

Compito

In un file word annota tutto quello che fai. Aggiungi anche gli screenshot per i punti principali (ad esempio copertura ...). Per effettuare gli screenshots usa lo strumento di cattura di windows. Anche la documentazione sarà valutata.

Sviluppa un progetto in eclipse per ognuna delle domande. Ogni progetto dovrà avere come nome il tuo COGNOME_**ES** (con ES il tipo dell'esercizio – vedi titoli degli esercizi). Puoi anche andare in parallelo (ad esempio modello NUSMV e codice Java). Metti i progetti e il documento word in una directory con nome COGNOME_NOME (che puoi usare come workspace). Alla fine crea un file zip e consegna quello.

Commenta per bene tutto il codice che scrivi.

Java

In Java scrivi una classe Magazzino che implementa il sistema sopra. Ha tre metodi:

boolean insert(int productIndex, int addQuantity)

dati un indice di prodotto productIndex e una quantita addQuantity, aggiunge il valore addQuantity (al massimo 10 prodotti); al prodotto identificato da productIndex Il metodo ritorna true se l'aggiunta viene eseguita, false altrimenti. L'aggiunta non viene eseguita se l'indice di prodotto è errato, o se la quantità aggiunta non è corretta (non compresa tra 1 e 10), o se la nuova quantità che si otterrebbe con l'aggiunta supera le 100 unita.

boolean isFull(int productIndex)

dice se un certo prodotto ha raggiunto il massimo

boolean isFull()

restituisce se il magazzino è completamente pieno (tutti i prodotti sono la massimo)

Le quantità di prodotti a magazzino sono memorizzate in un array di interi, dove ogni cella corrisponde ad un prodotto. All'inizio il magazzino è completamente vuoto.

1. ES = TESTING

Scrivere l'implementazione e i casi di test con Junit. Scrivi una classe di test FullTest per lo scenario in cui a partire il magazzino si riempie del tutto.

Esegui il controllo della copertura (istruzioni, branch, decisioni, MCDC). Se il caso sopra non è sufficiente ad avere una copertura soddisfacente, aggiungi i casi di test necessari (in una classe separata per ogni copertura).
Metti la directory test con i test separata (oltre src).

Prova con Randoop a generare casi di test e prova a valutarne la copertura.

2. ES = JML

Scrivi i contratti JML. Cerca di scrivere sia le precondizioni, che le postcondizioni dei metodi. Cerca di scrivere anche invarianti. Cerca di spostare nelle precondizioni alcuni controlli che facevi all'inizio dei metodi.
Aggiungi anche questi contatti:

- precondizione al metodo insert: esiste un prodotto con meno di 100 unita
- postcondizioni al metodo insert:
  - la media dei prodotti a magazzino e minore o uguale a 100;
  - la quantità dei prodotti diversi da productIndex è rimasta invariata.

Prova i contratti JML con una classe main in cui chiami i diversi metodi.

Prova anche a modificare il codice e controlla che i contratti siano violati. Documenta bene le violazioni e le loro cause in commenti e nel file di documento.

3. ES = KEY Verifica dei programmi

ATTENZIONE. Inizia da una classe con contratti **vuoti** (true) e **aggiungi** mano a mano i contratti però sempre dimostrandone la loro correttezza – cioè le prove devono essere sempre chiuse **vere**. La valutazione dipenderà dal massimo contratto che riuscite a dimostrare vero (contratti non provati non verranno valutati).

4. ES = NUMSV

Specificare il magazzino utilizzando NuSMV e le sue proprietà usando LTL e/o CTL.

Suggerimento: usa i moduli per rappresentare un singolo prodotto.

Puoi anche ammettere che più di un prodotto aumenti allo stesso tempo, però se riesci a far aumentare solo un prodotto tanto meglio.

Tra le proprietà desiderate (alcune potrebbero essere false) c'è:

- esiste uno stato in cui ci sono 100 unità di un prodotto;
- quando un prodotto è pieno, rimane pieno per sempre;
- il magazzino non è mai pieno.

Provare se sono vere oppure no. Se sono false presentare un contro esempio e modificarle in modo che siano vere.

Aggiungi anche altre proprietà che ti sembrano importanti e commenta.

Scrivi anche alcune proprietà sia di liveness che di safety e provale. Spiegale nel documento.

5. ES = FSM (MBT con le FSM)

Considera il caso semplificato di un magazzino con solo 2 prodotti in quantità da 0 a 5 a cui si possono aggiungere solo 1 o 2 prodotti. Modella l'evoluzione del sistema con una FSM. Implementa la FSM con modelJunit.

Salva il disegno della macchina come immagine.

6. ES = COMB (Combinatorial testing)

Rappresenta in combinatorial testing la chiamata del metodo boolean insert(int productIndex, int addQuantity)

Considera come variabili del problema:

productIndex e addQuantity che sono i parametri del metodo

returnedValue che il valore ritornato

nproductsOld che rappresenta il numero di prodotti prima della chiamata

nproductsNew che rappresenta il numero di prodotti dopo la chiamata

Introduci i vincoli opportuni. In questo modo hai anche l'oracolo.

Applica il combinatorial testing. Prova a generare la copertura pairwise. Prova a tradurre una riga dal tuo test in un test Junit per la classe Magazzino.