SAPIENZA Università di Roma
Facoltà di Ingegneria dell'Informazione, Informatica e Statistica
Corsi di Laurea in Ingegneria Informatica ed Automatica ed in Ingegneria dei Sistemi Informatici

Corso di Progettazione del Software

Esame del **22 gennaio 2015**

Tempo a disposizione: 3 ore

Requisiti. L'applicazione da progettare riguarda la gestione di un sistema di catene di montaggio per una azienda con i rispettivi nastri trasportatori. Un nastro trasportatore è caratterizzato da un insieme di pezzi di montaggio Ogni pezzo è caratterizzato da un colore (una stringa) ed un codice identificativo (una stringa) ed è situato su un solo nastro trasportatore. Alcuni pezzi sono "pesanti" e di questi interessa anche il peso. Al nastro trasportatore lavorano diversi robot multifunzione (almeno 2). Ciascun robot ha un nome (una stringa).

Un robot è inizialmente a riposo, e se riceve il messaggio di colorare con associato un nastro trasportatore, va nello stato di robot *verniciatore* mettendosi a lavorare per il nastro indicato, rimanendo in attesa. Similmente se riceve il messaggio di fissare il colore sempre con un nastro associato, va nello stato di robot *fissatore* mettendosi a lavorare per il nastro indicato, rimanendo in attesa. Se un robot nello stato *verniciatore* in attesa riceve un messaggio di verniciare con un pezzo associato, lo colora e lo manda ad un robot *fissatore* scelto casualmente tra quelli che lavorano allo stesso nastro[1], mettendosi in attesa che gli venga restituito. Quando un robot nello stato *fissatore* riceve un messaggio con il pezzo associato, ne fissa il colore, lo restituisce al robot *verniciatore* e si mette in attesa. Il robot *verniciatore* a cui vien restituito un pezzo fissato, gli da una seconda mano di vernice, lo rimette sul nastro trasportatore e si rimette in attesa. Sia nella configurazione *verniciatore* che nella configurazione *fissatore*, se il robot è in attesa e riceve il comando di resettarsi si rimette a riposo.

Siamo interessati alla seguente attività principale. L'attività prende in input un nastro trasportatore N e concorrentemente esegue le seguenti due sottoattività: *(i)* esegue la verniciatura dei pezzi, e *(ii)* l'analisi del nastro N. La sottoattività di verniciatura *(i)* avvia alcuni robot facendoli lavorare sul nastro N attraverso opportuni eventi (i dettagli non interessano). Poi si mette in attesa del comando di fine da parte dell'utente che fa terminare la sottoattività di verniciatura. La sottoattività di analisi *(ii)* calcola e stampa il numero di pezzi su ciascun nastro. Una volta che tali sottoattività sono state completate, si stampa un messaggio di saluto e si termina.

Domanda 1. Basandosi sui requisiti riportati sopra, effettuare l'analisi producendo lo schema concettuale in UML per l'applicazione, comprensivo del diagramma delle classi (inclusi vincoli non esprimibili in UML), diagramma stati e transizioni per la classe *Robot*, diagramma delle attività, specifica del diagramma stati e transizioni, e specifica dell'attività principale e delle sottoattività NON atomiche (indicando in modo esplicito quali attività atomiche sono di I/O e quali sono Task), motivando, qualora ce ne fosse bisogno, le scelte di progetto.

Domanda 2. Effettuare il progetto, illustrando i prodotti rilevanti di tale fase e motivando, qualora ce ne fosse bisogno, le scelte di progetto. È obbligatorio definire solo le responsabilità sulle associazioni del diagramma delle classi.

Domanda 3. Effettuare la realizzazione, producendo un programma JAVA e motivando, qualora ce ne fosse bisogno, le scelte di progetto. È obbligatorio realizzare in JAVA solo i seguenti aspetti dello schema concettuale:

- La classe `Robot`, con la classe `RobotFired`, e le classi JAVA per rappresentare le *associazioni* di cui `Robot` ha responsabilità.
- L'*attività principale* (NON le sue sottoattività).

[1] Dato un nastro, si assume che ci sia una funzione data che restituisce un robot *fissatore* tra quelli che lavorano a quel nastro.