SAPIENZA Università di Roma
Facoltà di Ingegneria dell'Informazione, Informatica e Statistica
Corsi di Laurea in Ingegneria Informatica ed Automatica
Corso di Progettazione del Software
Esame del **16 luglio 2024**
Tempo a disposizione: 3 ore

Requisiti. L'applicazione da progettare riguarda la gestione di robot agricoli all'interno di una vigna. Di ogni *robot* interessa il numero seriale (una stringa) ed il nome amichevole (una stringa). Esistono solo due tipi di robot: quelli raccoglitori e quelli logistici. Dei *robot raccoglitori* interessa il numero di braccia (un intero), mentre dei *robot logistici* interessa il numero di cassette che possono essere ospitate (un intero). All'interno della vigna lavorano degli *operatori umani* dei quali interessa il nome (una stringa) ed il cognome (una stringa). Durante la fase di raccolta dell'uva, un robot può essere associato ad un operatore ed è di interesse conoscere il timestamp (un intero lungo) in cui questa associazione avviene. Un operatore può essere associato ad un numero qualunque di robot. I robot logistici consegnano le cassette raccolte in appositi *magazzini* di cui si vuole conoscere il volume in metri cubi. Ogni robot logistico è associato ad almeno un magazzino e per ogni magazzino associato si vuole conoscere il numero di cassette consegnate. Un magazzino è invece associato ad un numero qualunque (anche nessuno) di robot logistici che sono però di interesse.

Siamo interessati al comportamento dei *robot raccoglitori* (nel seguito per semplicità `rr`). Un `rr` è inizialmente nello stato *ready*. In questo stato può ricevere un evento di *movimento* avente come parametri le coordinate x ed y del punto da raggiungere. In risposta a questo evento il `rr` va in stato *moving* avviando un'attività complessa di navigazione (che prende come parametri le coordinate x ed y ma di cui non interessano i dettagli) verso la posizione richiesta. Nello stato *moving*, il `rr` può ricevere evento di *fine movimento* che lo riporta nello stato *ready*. Sempre nello stato *ready*, il `rr` può ricevere dall'operatore umano associato un evento di *raccolta* riportante la direzione (che può assumere valori *left* o *right*). Questo evento porta il `rr` in stato *harvesting* avviando una attività complessa di raccolta (si vedano dopo i dettagli). Durante l'esecuzione dell'attività complessa, il `rr` può ricevere un evento di *errore* riportante una descrizione di tipo stringa che deve essere segnalato all'operatore che ha avviato la raccolta e che porta il `rr` in stato *manual*. Da questo stato, il `rr` può ricevere dall'operatore che ha avviato la raccolta o un evento di *continuazione* che riporta il `rr` in stato *harvesting*, oppure un evento di *interruzione* che porta il `rr` in stato *ready*. Nello stato di *harvesting*, il `rr` può ricevere un evento di *conclusione* che lo riporta in stato *ready*.

Siamo interessati all'attività complessa di *raccolta*. L'attività accetta come parametri un robot ed una direzione in cui effettuare la raccolta. Inizialmente l'attività complessa fa ruotare il robot verso la direzione richiesta. A questo punto si avviano due sotto-attività complesse in parallelo, una per le operazioni ed una che contiene una sola operazione atomica di raccolta log. L'attività di *operazioni* esegue ciclicamente le seguenti attività atomiche: *(i)* controlla la presenza di grappoli, *(ii)* sceglie il migliore di questi e *(iii)* lo raccoglie. I dettagli di queste attività atomiche non sono di interesse ma si scelgano opportuni input ed output. Se durante la raccolta di un grappolo si verifica un errore, un evento di *errore* è inviato al robot e l'attività si mette in attesa che un evento di *continuazione* o di *interruzione* venga inviato dall'operatore umano (si preveda un opportuno meccanismo[1]). Se non ci sono più grappoli da raccogliere, l'attività di *operazioni* si conclude inviando un evento di *conclusione* al robot. L'attività di *raccolta log* si conclude quando l'attività complessa di *operazioni* si conclude.

Domanda 1. Basandosi sui requisiti riportati sopra, effettuare l'analisi producendo lo schema concettuale in UML per l'applicazione, comprensivo di: diagramma delle classi (inclusi eventuali vincoli non esprimibili in UML); diagramma stati e transizioni per la classe *Robot Raccoglitore*; diagramma delle attività; specifica del diagramma stati e transizioni: segnatura dell'attività principale, sottoattività non atomiche, atomiche e segnali di input/output. Si noti che NON è richiesta la specifica delle attività. Motivare, 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. In particolare definire SOLO le responsabilità sulle associazioni del diagramma delle classi. Per le responsabilità utilizzare la seguente notazione: M per molteplicità, O per operazioni, ed R per requisiti.

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

[1] Ad esempio, si può implementare un meccanismo di attesa attiva sullo stato corrente del diagramma degli stati associato al robot.

- La classe *Robot* con le sue sottoclassi e la classe *RobotRaccoglitoreFired*, le classi Java per rappresentare l'associazione a responsabilità doppia tra *RobotLogistico* e *Magazzino*.
- L'attività *raccolta* e la sottoattivita' complessa di *operazioni*.

Nota: Rispetto a quanto sopra specificato, gli studenti con DSA *(i)* NON devono produrre la specifica del diagramma stati e transizioni, *(ii)* NON devono implementare l'attività principale di *raccolta*, ma solo la sottoattività di *operazioni*.