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 **12 settembre 2024**
Tempo a disposizione: 3 ore

Requisiti. L'applicazione da progettare riguarda la gestione di robot giardinieri da parte di una municipalità. La municipalità deve gestire un certo numero di giardini. Di ogni *giardino* interessa la via (una stringa) e l'area occupata in metri quadri (un numero reale). Ogni giardino è dotato di almeno un robot giardiniere. Di ogni *robot giardiniere* interessano il giardino (unico) in cui si trova ad il numero di serie (una stringa). Esistono solo due tipi di robot giardiniere, i falciatori e gli innaffiatori. Dei *falciatori* interessa la potenza (un numero reale), mentre degli *innaffiatori* interessa la capacità misurata in litri di acqua (un numero reale). Di ogni giardino interessano i programmi di manutenzione associati (almeno uno). Di ogni *programma di manutenzione* interessano la frequenza in ore (un intero) ed il percorso del file JSON che indica le impostazioni (una stringa). Un programma di manutenzione può essere associato ad un numero qualunque di giardini. La municipalità dispone inoltre di un certo numero di giardinieri umani (almeno uno). Di ogni *giardiniere umano* interessano i giardini a cui è assegnato (almeno uno), il codice fiscale (una stringa) ed il numero di cellulare (una stringa). Ogni giardino ha almeno un giardiniere umano.

Siamo interessati al comportamento dei *robot giardinieri* (nel seguito per semplicità `rg`). Un `rg` è inizialmente nello stato di *standby*. In questo stato può ricevere da un programma di manutenzione `sc` un messaggio di accensione, riportante come parametro il giardiniere umano `g` in carico di monitorare la procedura, che lo porta nello stato *stop*. Nello stato *stop* può ricevere da `sc` *(i)* un messaggio di spegnimento che lo riporta in stato *standby* oppure *(ii)* un messaggio di movimento avente come parametro una stringa che descrive il movimento stesso e che avvia un'attività complessa di movimento (di cui non interessano i dettagli) e che porta il `rg` in stato *movimento*. Da questo stato si esce perché si riceve dall'attività un messaggio di movimento concluso oppure un messaggio di errore. Entrambi i messaggi riportano il `rg` in stato *stop* notificando `sc`. Dopo tre messaggi di errore consecutivi però, invece di andare in stato *stop*, il robot va in stato *intervento manuale* notificando `g`. Dallo stato di *intervento manuale* si torna nello stato *stop* dopo che `g` invia un messaggio di sblocco, che viene notificato ad `sc`.

Siamo interessati all'attività complessa di *esecuzione di un programma di manutenzione*. L'attività accetta come parametro l'istanza di un programma di manutenzione `p` ed un giardino `giard`. Una volta avviata, l'attività esegue una prima attività atomica di scelta di un umano giardiniere `g` selezionato tra quelli associati a `giard`. A seguire, viene eseguita una attività atomica che seleziona la lista di robot manutentori da utilizzare (un sottoinsieme di quelli disponibili per `giard`) e per ognuno di essi calcola una lista di movimenti (si immagini una lista di stringhe) che rappresentano il piano in caso di mancanza di intoppi. Se n sono i robot selezionati, vengono avviati in parallelo n thread ognuno dei quali esegue l'attività complessa di *navigazione*. L'attività di *navigazione* prende in input un robot giardiniere, il corrispondente piano ed il giardiniere umano `g`. L'attività complessa di navigazione dopo avere inviato il messaggio di accensione al robot, implementa un ciclo che ad ogni iterazione esegue una attività atomica di selezione del prossimo passo (di cui non interessano i dettagli) che restituisce il prossimo passo da eseguire e lo invia come messaggio al robot giardiniere oppure termina l'attività se non ci sono più passi da eseguire inviando un messaggio di spegnimento al robot. Quando tutte le sottoattività complesse di navigazione sono terminate, si visualizza a schermo un messaggio di conclusione.

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 Giardiniere*; 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:

- La classe *RobotGiardiniere* con le sue sottoclassi e la classe *RobotGiardiniereFired*, le classi Java per rappresentare l'associazione a responsabilità doppia tra *RobotGiardiniere* e *Giardino*.
- L'attività complessa di *esecuzione programma di manutenzione* e la sottoattivita' complessa di *navigazione*.

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 *esecuzione programma di manutenzione*, ma solo la sottoattività complessa di *navigazione*.