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 **18 settembre 2023**
Tempo a disposizione: 2 ore e mezza

Requisiti.

L'applicazione da progettare riguarda la gestione del servizio di manutenzione strade di una città. Sono di interesse i *cantieri*, i *camion* e gli *operai*. Gli operai hanno nome, cognome, indirizzo e numero di telefono (una stringa). Alcuni operai sono anche *guidatori*, ma non tutti; dei guidatori interessa anche il numero della patente. Dei cantieri interessano le coordinate (latidudine, longitudine) e una descrizione (una stringa). Un operaio può essere assegnato a un camion o meno. Di ogni camion interessa la targa, il modello (due stringhe), gli eventuali operai assegnati (se ci sono) e l'eventuale cantiere a cui il camion è assegnato. Dato il cantiere, deve essere possibile cambiare i camion che sono stati assegnati ad esso. A un cantiere può anche non essere stato assegnato nessun camion.

Siamo interessati al comportamento dell'operaio. Un operaio è inizialmente a *riposo*. Quando riceve un evento *assengamento* con parametro un camion, passa nello stato *assegnato* senza aggiornare la sua associazione con i camion. Questo aggiornamento avviene solo quando riceve dall'autista del camion l'evento *arrivato*, che porta anche l'operaio nello stato *lavoro*. Questo evento viene inviato a tutti, per cui l'operaio deve verificare che il mittente sia assegnato allo stesso suo camion. Nello stato ~~*arrivato*~~ *lavoro* può ricevere eventi *aggiornamento*, a cui risponde con il risultato di un suo metodo `String avanzamento()`, di cui non occorre fornire implementazione. Se il risultato di questo metodo è la stringa `"terminato"`, passa nello stato *riposo*.

Siamo interessati alla seguente attività che prende come parametri un camion e un insieme di operai. L'attività principale cerca un operaio che non sia attualmente assegnato a nessun camion. Se non ci riesce, emette un messaggio di errore e termina. Se invece ci riesce, lancia due sottoattività parallele di monitorizzazione dello stato e di compilazione di un report, di cui interessa solo il fatto che la seconda produce una stringa, che viene stampata quando entrambe le sottoattività parallele terminano.

Domanda 1. Basandosi sui requisiti riportati sopra, effettuare la fase di 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 *Operaio*, diagramma delle attività, la segnatura delle attività complesse, delle attività atomiche e dei segnali di input/output. Motivare, qualora ce ne fosse bisogno, le scelte di progetto.

Domanda 2. Effettuare la fase di progetto, illustrando i prodotti rilevanti di tale fase e motivando, qualora ce ne fosse bisogno, le scelte effettuate. In particolare definire SOLO le responsabilità sulle associazioni del diagramma delle classi. Nella tabella, inserire anche il motivo di ognuna delle responsabilità usando le lettere M, O, R rispettivamente per molteplicità, operazioni e requisiti.

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

- La classe *Operaio* e le sue eventuali *sottoclassi*, le classi che rappresentano *l'associazione* a responsabilità doppia fra *Operaio* e *Camion*, i metodi della classe *Camion* di questa associazione e la classe *OperaioFired*.
- L'*attività principale* e l'attività *atomica* che verifica se un operaio è assegnato o meno a un camion.