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 luglio 2023**
Tempo a disposizione: 3 ore

Requisiti. L'applicazione da progettare riguarda la gestione da parte della polizia locale della rete stradale cittadina intelligente. Sono di interesse gli incroci cittadini. Ogni *incrocio* ha un codice identificativo (una stringa), latitudine, longitudine ed un insieme di stringhe che rappresenta le strade connesse dell'incrocio. Di un incrocio sono di interesse inoltre i semafori (almeno uno). Di ogni *semaforo* sono di interesse l'incrocio presso cui è installato, il codice identificativo (una stringa), la compagnia manutentrice (una stringa), la data di costruzione e l'eventuale colore corrente ("verde", "giallo", "rosso", "giallo lampeggiante"). Inoltre, si vogliono conoscere ad ogni istante i mezzi di trasporto fermi ad ogni semaforo (eventualmente nessuno). Di ogni *mezzo di trasporto* interessano la targa (una stringa), la marca, il modello e l'eventuale semaforo a cui si è fermato. I mezzi di trasporto si distinguono in mezzi corti e lunghi. Dei *mezzi corti* (es. le automobili) interessa la cilindrata, mentre dei *mezzi lunghi* interessa il peso.

Siamo interessati al comportamento dei *semafori*. Inizialmente un semaforo è *spento*. Nello stato spento, esso può ricevere un evento di *accensione* che contiene una soglia di criticità (un intero) ed oggetto di tipo coda il cui uso è spiegato dopo. Una volta ricevuto l'evento di accensione, il semaforo si sposta nello stato *acceso* colorandosi di "rosso". Nello stato *acceso* può ricevere dai mezzi di trasporto eventi di *avvicinamento* ed *allontanamento* in seguito ai quali viene aggiornata l'associazione tra *semaforo* e *mezzo di trasporto*. Se il numero di mezzi di trasporto al semaforo supera la soglia di criticità, il semaforo va in stato *critico*, scrivendo sulla coda una stringa formata dal codice del semaforo seguito da "CRITICO" (come scrivere sulla coda è spiegato dopo). Nello stato *critico* il semaforo continua a ricevere eventi di *avvicinamento* ed *allontanamento*. Se il numero di mezzi di trasporto al semaforo va sotto la soglia di criticità, il semaforo torna in stato *acceso* aggiungendo sulla coda una stringa formata dal codice del semaforo seguito da "NON CRITICO". Nello stato *acceso* ed in quello *critico* il semaforo può inoltre ricevere un evento di *spegnimento* che riporta il semaforo in stato spento ed il colore dello stesso in "giallo lampeggiante".

Siamo interessati all'attività principale di gestione di un incrocio. L'attività è inizializzata con un'istanza di incrocio. L'attività chiede all'utente di impostare una soglia critica ed in seguito crea la coda per la comunicazione tra i semafori. A questo punto viene chiamata una attività che invia segnali di accensione a tutti i semafori dell'incrocio. Si aprono a questo punto due sotto-attività complesse. La prima sotto-attività complessa si mette semplicemente in attesa di un comando di spegnimento, ricevuto il quale un comando "FINE" viene scritto sulla coda. La seconda sotto-attività si mette in attesa di messaggi sulla coda. Non appena un messaggio viene ricevuto, se il messaggio ricevuto è "FINE", si termina la sotto-attività, altrimenti si avvia una attività atomica di reimpostazione dei semafori (di cui non interessano i dettagli) e si ritorna in attesa di nuovi messaggi. Una volta che entrambe le sotto-attività complesse si sono completate, un evento di spegnimento viene inviato a tutti i semafori dell'incrocio.

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 *Semaforo*; 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[1] SOLO i seguenti aspetti dello schema concettuale:

- La classe *Semaforo* con la classe *SemaforoFired*, le classi Java per rappresentare l'associazione a responsabilità doppia tra *Semaforo* e *MezzoDiTrasporto*.

[1] Si realizzi la coda di comunicazione tra semafori ed attività utilizzando un'istanza della classe `ArrayBlockingQueue<String>` che è thread safe e non richiede sincronizzazione. In particolare, utilizzare il metodo `put(String s)` per inserire elementi nella coda ed il metodo `String take()` per mettersi in attesa di nuovi messaggi.

- L'attività principale e le due sottoattivita' complesse che vengono lanciate in parallelo dall'attivita' principale. NON si richiede invece l'implementazione delle attività atomiche.

Nota: Rispetto a quanto sopra specificato, gli studenti con DSA *(i)* NON devono produrre la specifica del diagramma stati e transizioni, *(ii)* possono realizzare solo l'attività principale e la sottoattività complessa che riceve messaggi dai semafori.