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 **23 gennaio 2023**
Tempo a disposizione: 3 ore

Requisiti. L'applicazione da progettare riguarda la gestione da parte di una azienda di trasporto pubblico dei mezzi a disposizione e delle linee di trasporto. Di ogni *mezzo pubblico* interessa conoscere la matricola (una stringa) ed il numero di posti a sedere. Esistono solo due tipi di mezzo pubblico: gli *autobus* ed i *treni*. Degli autobus interessa il numero di porte, mentre dei treni interessa il numero di vagoni. Sono inoltre di interesse le *linee* del trasporto pubblico. Di ogni linea, interessa il nome (una stringa, ad esempio "163"), le *paline* (almeno due ed in sequenza ordinata) di cui è composta ed i mezzi pubblici (almeno uno) assegnati a quella linea. Di ogni palina interessa conoscere un codice (un intero), la latitudine (un reale), la longitudine (un reale) e l'indirizzo (una stringa). Ogni palina fa parte di almeno una linea. Alcune paline sono *capolinea*. Dei capolinea interessa la superficie in metri quadri dell'area occupata (un reale). Per ogni autobus è inoltre di interesse memorizzare gli ultimi passaggi alle paline (non è di interesse lo storico), ognuna con un timestamp associato (un intero lungo). Si noti che per ogni coppia (autobus, palina) è di interesse memorizzare solo l'ultimo passaggio.

Siamo interessati al comportamento degli *autobus*. Un autobus è inizialmente nello stato *spento*. Da questo stato, una volta ricevuto il comando di accensione (con parametro il nome dell'autista), passa allo stato *acceso*. La transizione in stato acceso ha come effetto l'avvio dell'attività principale di AVM (descritta nel paragrafo successivo), passando come parametro l'autobus ed il nome dell'autista. Nello stato acceso, l'autobus può ricevere l'evento di spegnimento che lo riporta nello stato spento, stampando a schermo un saluto riportante il nome di chi ha acceso il veicolo. Inoltre nello stato accesso, l'evento di innesto della modalità folle porta il veicolo in stato *stop*. Dallo stato stop, la ricezione dell'evento di innesto della modalità drive riporta il veicolo in stato avvio.

Siamo interessati all'attività principale di AVM - Automatic Vehicle Monitoring che viene avviata al passaggio dell'autobus in stato *avvio*. L'attività prende come parametro di ingresso l'istanza dell'autobus su cui l'AVM gira ed il nome dell'autista. Una volta acceso, l'AVM mostra un messaggio di saluto riportante il nome dell'autista (su uno schermo in dotazione) e gli chiede di inserire il numero della linea. A questo punto, tramite l'attività atomica `verificaLinea`, l'AVM verifica che linea sia tra quelle assegnate all'autobus. In caso negativo, ritorna a chiedere all'autista il numero di linea, altrimenti prosegue nell'attività. A questo punto si attivano due sottoattività complesse in parallelo. La prima sottoattività complessa è costituita da una attività atomica `aggiornaPosizione` che confronta la posizione corrente (i dettagli su come questa posizione viene ottenuta non sono di interesse) dell'autobus con quella di ogni palina sulla linea selezionata correntemente e, se l'autobus si trova nei pressi di una palina, aggiorna di conseguenza il timestamp di ultima visita. Alla fine di ogni esecuzione di `aggiornaPosizione` la sottoattività complessa controlla lo stato corrente dell'autobus, e se questo è *spento* termina la sottoattività, altrimenti esegue di nuovo `aggiornaPosizione`. La seconda sottoattività è similare alla prima, dove l'attività atomica è `controllaSaluteAutobus`, i cui dettagli non sono di interesse, che effettua dei controlli sullo stato di salute dell'autobus. Una volta che entrambe le sottoattività terminano, anche l'attività principale termina.

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 *Autobus*; 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 *MezzoPubblico* (ed eventuali sottoclassi) con la classe *AutobusFired*, le classi Java per rappresentare le associazioni di cui la classe *MezzoPubblico*, o eventuali sottoclassi, hanno responsabilità.

- L'attività principale e le sue eventuali sottoattività NON 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 una sola delle associazioni su cui la classe *MezzoPubblico* o eventuali sottoclassi hanno responsabilità, *(iii)* possono realizzare solo l'attività principale ed una delle sottoattività complesse.