SAPIENZA Università di Roma
Facoltà di Ingegneria dell'Informazione, Informatica e Statistica
Corsi di Laurea in Ingegneria Informatica ed Automatica ed in Ingegneria dei Sistemi Informatici

Corso di Progettazione del Software

Esame del **11 Sett embre 2018**

Tempo a disposizione: 3 ore

Requisiti. L'applicazione da progettare riguarda la gestione di carovane per attraversare il deserto. Una carovana ha un nome (una stringa) ed è costituita da uno o più mezzi (di trasporto persone). Ogni mezzo e di un determinato tipo (una stringa) e trasporta uno o più persone. Ogni persona ha un nome (una stringa) ed è trasportato da esattamente un mezzo. I mezzi sono suddivisi in mezzi a guida autonoma e mezzi a guida non autonoma. Dei primi interessa l'anno dell'ultimo collaudo (in intero). Dei secondi interessa chi tra le proprie persone trasportate è il pilota, con il numero di volte che egli ha guidato il mezzo stesso.

Siamo interessati al comportamento dei mezzi. Un mezzo è inizialmente alla *a riposo*. Se *a riposo* riceve l'evento partenza si mette *in marcia* (con le proprie persone incluso il pilota). Se *in marcia* riceve il evento di *carica*, dalla carovana o da un'altro mezzo, con payload un insieme di utenti, li aggiunge alle proprie persone trasportate, rilanciando l'evento *caricate* al mittente dell'evento *carica*, rimanendo *in marcia*. Se *in marcia* riceve l'evento *guasto* manda in broadcasting la richiesta di *accogliere* le proprie persone e si mette in *attesa*. Se quando in *attesa* riceve l'*ok* da un'altro mezzo, manda ad esso l'evento *carica* con payload l'intero insieme delle proprie persone escluso il pilota mettendosi *in trasbordo*. Quando in *in trasbordo* riceve l'evento *caricate* del mezzo scelto si mette a *riposo*. Infine se *in marcia* riceve l'evento *stop* si mette a *riposo*.

Siamo interessati alla seguente attività principale. L'attività prende in input una carovana C ed un numero positivo n e verifica che il numero di utenti in ogni mezzo della carovana C sia inferiore o uguale a n. Se la verifica non va a buon fine, l'attività termina segnalando in output un errore. Altrimenti concorrentemente esegue le seguenti due sottoattività: *(i)* esecuzione, e *(ii)* analisi. La sottoattività di esecuzione *(i)* avvia l'esecuzione di tutti i mezzi mandando opportuni eventi (i dettagli non interessano) e si mette in attesa del segnale di input di fine-esecuzione che interrompe l'esecuzione stessa. La sottoattività di analisi *(ii)* calcola la percentuale di mezzi autonomi sul numero totale di mezzi della carovana C. Una volta che tali sottoattività sono state completate, l'attività principale manda un segnale di output con il risultato dell'attività di analisi e termina.

Domanda 1. Basandosi sui requisiti riportati sopra, effettuare l'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 *Mezzo*, diagramma delle attività, specifica del diagramma stati e transizioni, e specifica dell'attività principale e delle sottoattività NON atomiche (indicando in modo esplicito quali attività atomiche sono di I/O e quali sono Task), motivando, 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. È obbligatorio definire solo le responsabilità sulle associazioni del diagramma delle classi.

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

- La classe `Mezzo` con classe `MezzoFired`, le eventuali sottoclassi, e le classi JAVA per rappresentare le *associazioni* di cui la classe `Mezzo` e le sue sottoclassi hanno responsabilità.
- L'*attività principale* e le sue eventuali sottoattività NON atomiche.