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 **14 Giugno 2018**

Tempo a disposizione: 3 ore

Requisiti. L'applicazione da progettare riguarda una parte di un videogioco di fantascienza. Un gioco è costituito da diversi giocatori. I giocatori, che hanno un nome (una stringa) sono divisi in 3 categorie: i piloti, i droidi e i passeggeri. Dei piloti interessano le ore di volo (un intero), dei droidi il modello (una stringa) e dei passeggeri il peso trasportato (un reale). Ogni pilota è abilitato a pilotare un insieme non vuoto di astronavi. Delle astronavi interessa la descrizione (una stringa). Ai piloti piace volare con alcuni droidi (almeno uno) e tra questi uno in particolare che funge da compagno di volo. Ogni droide è il compagno di volo di esattamente un pilota. Altri dettagli non interessano.

Siamo interessati al comportamento dei piloti. Un pilota è inizialmente alla *base*. Se riceve il comando di *volare* con payload una astronave e un passeggero, se il pilota è abilitato a pilotare l'astronave allora chiede al suo droide compagno di salire a bordo dell'astronave, mettendosi in *attesa del droide*. Quando questo gli comunica *droide-salito* allora chiede al passeggero di salire a bordo dell'astronave, mettendosi in *attesa del passeggero*. Quando il passeggero gli comunica *passeggero-salito* si mette *in volo*. Cosa fa quando in volo non interessa, eccetto che quando riceve il comando di *rientro* torna alla *base*. Il comportamento degli altri tipi di giocatore non interessa.

Siamo interessati alla seguente attività principale. L'attività prende in input un gioco G verifica che il numero di piloti, droidi e passeggeri di G sia congruo e che tutti i droidi compagni dei piloti di G siano anche loro giocatori di G (i dettali non interessano). Se la verifica non va a buon fine, l'attività termina segnalando in output un errore. Altrimenti concorrentemente esegue le seguenti due sottoattività: *(i)* gioca, e *(ii)* analisi. La sottoattività di gioco *(i)* avvia il gioco attivando tutti i giocatori di G mandando opportuni eventi (i dettagli non interessano). Poi si mette in attesa del comando di fine-gioco da parte dell'utente che interrompe il gioco. La sottoattività di analisi *(ii)* calcola quanti piloti, droidi e passeggeri sono presenti nel gioco G mandando un report con questi dati in output. Una volta che tali sottoattività sono state completate, l'attività principale manda un segnale di output di saluto 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 *Pilota*, 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 `Pilota` con classe `PilotaFired`, le eventuali superclassi, e le classi JAVA per rappresentare le *associazioni* di cui la classe `Pilota` ha responsabilità.
- L'*attività principale* e le sue eventuali sottoattività NON atomiche.