SAPIENZA Università di Roma
Facoltà di Ingegneria dell'Informazione, Informatica e Statistica
Corso di Laurea in Ingegneria Informatica e Automatica
Corso di Laurea in Ingegneria dei Sistemi Informatici
Esame di Progettazione del Software
Appello del **21 febbraio 2012**
Tempo a disposizione: 3 ore

Requisiti. L'applicazione da progettare riguarda una variante del gioco della laser-tag. Una partita è caratterizzata da un nome e da un insieme di squadre che partecipano al gioco (la cardinalità di detto insieme deve essere maggiore o uguale a 2). Ciascuna squadra partecipa ad una sola partita ed è caratterizzata da un intero che indica il colore. Ogni squadra contiene un insieme ordinato non vuoto di giocatori. Ogni giocatore partecipa esattamente ad una squadra ed è caratterizzato da un avatar (una stringa che rappresenta il path al file `.jpg`) da un intero compreso tra 0 e 99 che rappresenta la capacità di schivare colpi, da un secondo intero anche esso compreso tra 0 e 99 che rappresenta la mira del giocatore stesso, e da un terzo intero che rappresenta il numero di colpi ricevuti durante la partita stessa.

Un giocatore è inizialmente "non in gioco". Quando riceve l'evento "inizio partita", azzera il numero di volte in cui è stato colpito e passa allo stato "corri" che mantiene fino a quando riceve l'evento "spara" generato da se stesso o "colpo" da un giocatore qualsiasi. Quando riceve uno di questi eventi passa alla stato "fermo" ma producendo due azioni diverse. Se riceve l'evento "spara", seleziona un giocatore di una qualsiasi squadra diversa dalla sua, secondo una funzione che si assume data, gli spara un colpo (generando un evento "colpo") e lancia un evento "scappa" a se stesso. Se riceve un "colpo", incrementa o meno il numero di colpi subiti secondo una funzione booleana (che si assume data) che calcola un valore random sulla base della mira del giocatore che ha sparato e della capacità di schivare del giocatore che subisce il colpo, e lancia un evento "scappa" a se stesso. Quando è nello stato "fermo" torna nello stato "corri" ricevendo un evento "scappa" generato da se stesso. Quando un giocatore riceve l'evento di "fine partita" torna nello stato "non in gioco". Eccetto che per i colpi subiti, il giocatore può essere modificato solo quando è nello stato "non in gioco".

Siamo interessati a progettare l'attività del gioco, che prende come parametro una partita, corredata di squadre e giocatori e procede eseguendo concorrentemente le seguenti sottoattività: *(i)* inizia il gioco, attraverso l'invio in broadcasting a tutti i giocatori dell'evento "inizio partita"; *(ii)* si mette in attesa (attraverso un'opportuna attività di input/output) del comando di fine esecuzione da parte dell'utente, che termina il gioco riportando tutti i giocatori nello stato "non in gioco". Una volta che tali sottoattività sono completate, calcola il numero di colpi subiti da ciascuna squadra e stampa la squadra vincitrice (o le squadre vincitrici) cioè quella che ha subito meno colpi.

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 *Giocatore*, diagramma delle attività, specifica del diagramma stati e transizioni, e specifica della attività principale (indicando in modo esplicito quali attività atomiche sono di I/O e quali sono Task), motivando qualora ce ne fosse bisogno le scelte effettuate.

Domanda 2. Effettuare la fase di progetto, illustrando i prodotti rilevanti di tale fase e motivando, qualora ce ne fosse bisogno, le scelte effettuate. È obbligatorio definire solo le responsabilità sulle associazioni del diagramma delle classi.

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 *Giocatore* (con eventuale classe *GiocatoreFired*, ma senza sottoclassi) e le eventuali *associazioni* che la legano alla classe *Squadra*.
- L'*attività principale*, e le eventuali *sottoattività NON atomiche*.