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 **22 gennaio 2013**

Tempo a disposizione: 3 ore

Requisiti. L'applicazione da progettare riguarda una versione modificata del *paintball tutti contro tutti*. I giocatori sono suddivisi in squadre di diversi colori. Un gioco è caratterizzato da un nome e da un numero arbitrario, ma superiore a 3, di squadre. Ad un gioco partecipano un insieme ordinato, di cardinalità superiore a 30, di giocatori. Le squadre possono essere alleate tra loro (si noti che "essere alleato" è una relazione simmetrica: se x è alleato a y allora y è alleato a x). Ogni giocatore che partecipa al gioco appartiene esattamente ad una squadra. Un giocatore ha un nome ed un avatar (una stringa che rappresenta il path al file `.jpg`). Per ciascun giocatore è di interesse ricordare quante volte è stato colpito durante l'esecuzione del gioco da giocatori di ciascuna squadra separatamente.

Un giocatore è inizialmente *non in gioco*. Quando riceve l'evento *inizio partita* passa allo stato *in gioco*. Ciascun giocatore in gioco, quando riceve un evento *colpo*, verifica a quale squadra appartiene chi ha lanciato il colpo e incrementa il numero di colpi subiti da detta squadra; poi rilancia un evento *colpo* ad un altro giocatore del gioco scelto arbitrariamente. Quando un giocatore riceve l'evento *fine partita* il giocatore torna nello stato *non in gioco*. Il giocatore può essere modificato solo quando è in *non in gioco*.

Siamo interessati a progettare l'attività del gioco, che prende come parametro un gioco completo delle squadre e dei loro giocatori. Viene prima stampato un messaggio di benvenuto. Quindi, contemporaneamente, si procede a: *(i)* giocare, attraverso l'invio in broadcasting a tutti i giocatori dell'evento *inizio partita* e dell'ulteriore invio dell'evento *colpo* a 10 giocatori scelti arbitrariamente (tale evento *colpo* ha come mittente i giocatori stessi che lo subiscono), mettendosi poi 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"; *(ii)* si calcola e poi si stampa un report su tutte le squadre, elencando i loro nomi e i nomi dei loro giocatori. Una volta terminate entrambe le sottoattività, si verifica quali sono le squadre i cui giocatori hanno ricevuto complessivamente meno colpi, ignorando i colpi subiti da squadre alleate, e le si eleggono vincitrici. L'attività si conclude stampando le squadre vincitrici.

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, diagramma stati e transizioni per la classe *Giocatore*, diagramma delle attività, specifica del diagramma stati e transizioni, e specifica delle attività complesse, 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:

- Le classi *Giocatore* (con le eventuali classi *fired* corrispondenti) e le *associazioni* che le legano alla classe *Squadra*.
- L'*attività principale* con eventuali sottoattività, escluse le attività atomiche.