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

Corso di Progettazione del Software

Esame del **19 Luglio 2010**

Tempo a disposizione: 3 ore

Requisiti. L'applicazione da progettare riguarda una variante del gioco della bottiglia. Un partita è caratterizzata da un nome, un insieme ordinato di partecipanti ed un insieme non vuoto ordinato di penitenze. Ogni partecipante partecipa esattamente ad una partita ed è caratterizzato da un nome, da un avatar (una stringa che rappresenta il path al file `.jpg`) e dall'insieme delle penitenze che ha subito durante la partita, ciascuna con il numero di volte che è stata subita. I partecipanti sono suddivisi in"ragazzi" e "ragazze". Una penitenza è caratterizzata da una descrizione testuale della stessa (una stringa).

Un partecipante è inizialmente "non in gioco". Quando riceve l'evento "inizio partita", azzera le penitenze subite e passa allo stato "in gioco". Se un partecipante "ragazzo" è in gioco e viene indicato dalla bottiglia (riceve l'evento "bottiglia"), allora se il mittente è un "ragazzo" oppure "non-noto" rilancia semplicemente la bottiglia indicando a caso (in modo random) uno tra i partecipanti alla partita. Se invece il mittente è una "ragazza" allora, prima di rilanciare la bottiglia come nel caso precedente, deve selezionare in modo casuale una delle penitenze della partita e aggiungerla al suo insieme di penitenze subite. I partecipanti "ragazza" si comportano in maniera analoga, subendo penitenze solo se ricevono la bottiglia da un "ragazzo". Quando un partecipante riceve l'evento di "fine partita" torna nello stato "non in gioco". Eccetto che per le penitenze subite, il partecipante può essere modificato solo quando è in "non in gioco".

Siamo interessati a progettare l'attività del gioco, che prende come parametro una partita, in cui è già definito il nome e l'insieme delle penitenze, ma non i partecipanti. L'attività inizia leggendo in input l'insieme (non vuoto) di partecipanti, dopo di che, li inserisce nella partita. A questo punto verifica la giocabilità della partita: cioè se l'insieme dei partecipanti contiene almeno un partecipante "ragazzo" ed almeno un partecipante "ragazza". Se la verifica da un esito negativo allora termina visualizzando un messaggio d'errore, altrimenti procede eseguendo concorrentemente le seguenti sottoattività: *(i)* inizia il gioco, attraverso l'invio in broadcasting a tutti i partecipanti dell evento "inizio partita" e dell'ulteriore invio dell'evento"bottiglia" (con mittente "non-noto") ad un partecipante qualsiasi; *(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, si visualizza un messaggio di saluto e l'attività principale termina.

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 *Partecipante*, diagramma delle attività, specifica del diagramma stati e transizioni, specifica della attività principale (indicando in modo esplicito quali attività atomiche sono di I/O e quali sono Task) e specifica della attività atomica di verifica di giocabilità, 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 *Partecipante* (con eventuale classe *PartecipanteFired*, ma senza sottoclassi) e le eventuali *associazioni* che la legano alla classe *Partita*.
- L'*attività principale*, e le *sottoattività non atomiche*.