SAPIENZA Università di Roma
Facoltà di Ingegneria – Corsi di Laurea in Ingegneria Informatica e Automatica
e in Ingegneria dei Sistemi Informatici
Corso di Progettazione del Software
Esame del **Esempio di testo d'esame**
Tempo a disposizione: 3 ore

Requisiti. L'applicazione da progettare riguarda una variante del gioco dell'oca in cui si inizia in caselle qualsiasi (decise in modo random) e si procede a turno lanciando dati. La partita termina quando un giocatore arriva per primo in una casella senza successore.

Più precisamente, ogni casella ha un disegno (una stringa che denota il file jpg) ed ha al più una casella "successiva". Alcune caselle sono speciali caselle "salto" che hanno un nome e indicano oltre alla eventuale casella successiva anche la casella in cui si salta. Le caselle possono essere occupate dai giocatori, ogni giocatore è caratterizzato dal nome (una stringa) e da un "avatar" (una piccola immagine rappresentata da una stringa che denota il file "animated jpg" che si giustappone al disegno della casella), e occupa al più una casella. Una partita è caratterizzata da un nome e da un insieme ordinato di almeno due giocatori (l'ordine indica la successione con cui il lancio del dado spetta al successivo giocatore).

Un giocatore è inizialmente "in allenamento". Quando riceve l'evento "inizio partita" passa allo stato "in partita". Ciascun giocatore in partita, quando è il suo turno (segnalato dall'evento "lancia dado") percorre un numero di caselle pari al valore uscito nel lancio del dado (il numero effettivo è random); se la casella in cui arriva è di "salto", allora si muove nella casella in cui si salta (ripetendo la cosa anche più volte se la casella d'arrivo è essa stessa di salto). Se tra le caselle che percorre ne incontra una senza successore (ha vinto la partita), il giocatore lancia un evento di "fine partita" a tutti i giocatori (incluso se stesso) , altrimenti passa il turno al giocatore successivo, attraverso un nuovo evento "lancia dado". Quando riceve l'evento di "fine partita" il giocatore torna in allenamento. Il giocatore può essere modificato solo quando è in allenamento.

Siamo interessati a progettare l'attività di gioco, che prende come parametro un insieme di caselle (il tabellone) e procede verificando inizialmente che il tabellone sia valido, cioè soddisfa le seguenti tre condizioni: *(a)* contiene almeno una casella "normale"; *(b)* ogni casella ha come casella "successiva" una casella appartenente al tabellone; *(c)* ogni casella "salto" ha come casella in cui si salta una casella ancora appartenete al tabellone. Se il tabellone non è valido si termina. Altrimenti si leggono input n giocatori, si crea una partita con il nome "partita corrente" associando i giocatori alla stessa e posizionando in modo casuale i giocatori in caselle con un successore. 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"lancio dado" al primo giocatore della partita; *(ii)* mettersi in attesa (attraverso un'opportuna attività di input/output) del comando di fine esecuzione da parte dell'utente.

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 diagrammi stati e transizioni, specifica delle attività complesse e specifica della attività atomica di verifica di validità del tabellone , 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* e le eventuali *associazioni* che la legano alla classe *Casella*.
- L'*attività principale*, l'*attività atomica di verifica della validità del tabellone*.